



COMPUTE!'s THIRD BOOK OF APPLE

Our best applications, games, tutorials, and programming utilities for the Apple II, II+, IIe, and IIfx computers. Powerful spreadsheet, high-energy arcade games, comprehensive financial tools, and more.

A **COMPUTE! Books** Publication

\$14.95

COMPUTE!'s THIRD BOOK OF
APPLE

COMPUTE!
Books



COMPUTE!'s THIRD BOOK OF APPLE

COMPUTE! Publications, Inc. 

Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies

Greensboro, North Carolina

The following articles were originally published in *COMPUTE!* magazine, copyright 1985, COMPUTE! Publications, Inc.: "Mousor" (August); "Easy Apple Screen Editing" (September); "Apple Pull-Down Menus" (October); "The Witching Hour" (October); "Puzzler" (November); "Skyscape" (November); "Apple Disk Booster" (December); "Apple Hi-Res Screen Dump" (December); "Memo Diary" (December).

The following articles were originally published in *COMPUTE!'s Apple Applications Special*, copyright 1985, COMPUTE! Publications, Inc.: "The ApWriter" (Fall/Winter); "Dr. Disk" (Fall/Winter); "Apple Animator" (Fall/Winter).

The following articles were originally published in *COMPUTE!* magazine, copyright 1986, COMPUTE! Publications, Inc.: "Apple Keyboard Customizer" (January); "Instant Apple Help Screens" (February); *SpeedCalc* (February); "High Rise" (February); "Mousify Your Applesoft Programs" (March/April); "MultiMemory" (March); "Switchbox" (March); "Apple Disk Duper" (April); "Hi-Res Graphics Aid Routines" (April); "Tug-A-War" (April); "Applesoft List Enhancer" (May); "Hickory, Dickory, Dock" (May); "Miami Ice" (June).

The following articles were originally published in *COMPUTE!'s Apple Applications Special*, copyright 1986, COMPUTE! Publications, Inc.: "Keynote" (Spring/Summer); "Your Personal Ledger" (Spring/Summer); "Windows" (Spring/Summer).

Copyright 1986, COMPUTE! Publications, Inc. All rights reserved.

Reproduction or translation of any part of this work beyond that permitted by Sections 107 and 108 of the United States Copyright Act without the permission of the copyright owner is unlawful.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-87455-063-7

The authors and publisher have made every effort in the preparation of this book to insure the accuracy of the programs and information. However, the information and programs in this book are sold without warranty, either express or implied. Neither the authors nor COMPUTE! Publications, Inc., will be liable for any damages caused or alleged to be caused directly, indirectly, incidentally, or consequentially by the programs or information in this book.

The opinions expressed in this book are solely those of the authors and are not necessarily those of COMPUTE! Publications, Inc.

COMPUTE! Publications, Inc., Post Office Box 5406, Greensboro, NC 27403, (919) 275-9809, is part of ABC Consumer Magazines, Inc., one of the ABC Publishing Companies, and is not associated with any manufacturer of personal computers. Apple, Apple II, Apple II+, Apple IIe, Apple IIc, DOS 3.3, and ProDOS are trademarks of Apple Computer, Inc.

Contents

Foreword	v
Chapter 1. Recreation	1
Tug-A-War	
Mark Tuttle (<i>Apple version by Tim Victor</i>)	3
Switchbox	
Todd Heimarck (<i>Apple version by Tim Victor</i>)	7
High Rise	
Charles McGuyer (<i>Apple version by Tim Victor</i>)	18
The Witching Hour	
Brian Flynn (<i>Apple version by Kevin Martin</i>)	33
Miami Ice	
Jeff Kulczycki (<i>Apple version by Tim Victor</i>)	39
Chapter 2. Education	51
Hickory, Dickory, Dock	
Barbara H. Schulak (<i>Apple version by Tim Victor</i>)	53
Skyscape	
Robert M. Simons (<i>Apple version by Tim Victor</i>)	57
Puzzler	
Mark Tuttle and Kevin Mykytyn	
(<i>Apple version by Kevin Martin</i>)	70
Chapter 3. Applications	75
Apple SpeedCalc	
Kevin Martin	77
Memo Diary	
Jim Butterfield	133
The ApWriter	
Tim Victor	144
Your Personal Ledger	
Alan H. Stein	153
Dr. Disk	
Alan H. Stein	173
Chapter 4. Graphics	199
Apple Animator	
Steve Johnson (<i>Apple version by Tim Victor</i>)	201
Hi-Res Graphics Aid Routines	
Jon Hylands	216

Apple Hi-Res Screen Dump	
<i>Mark Russinovich</i>	222
Chapter 5. A New Apple II	227
Windows	
<i>Lee Swoboda</i>	229
Apple II Pull-Down Menus	
<i>Lee Swoboda</i>	241
Mousify Your Applesoft Programs	
<i>Lee Swoboda</i>	250
Mousor: Escape Mode Cursor for the Apple IIc	
<i>J. Blake Lambert and Tim Victor</i>	275
Chapter 6. Programming and Utilities	281
Keynote	
<i>Patrick Parrish</i>	283
Easy Apple Screen Editing	
<i>Roland Brown (Enhanced version by Tim Victor)</i>	286
Apple Disk Booster	
<i>D. W. Hoover</i>	292
Apple Keyboard Customizer	
<i>Robert Buehler</i>	296
Instant Apple Help Screens	
<i>Kent Brewster</i>	302
MultiMemory	
<i>Patrick Parrish</i>	308
Apple Disk Duper	
<i>Jason Coleman</i>	313
Applesoft List Enhancer	
<i>Steven Roth</i>	315
Appendices	317
A. Guide to Typing In Programs	319
B. Apple Automatic Proofreader	
<i>Tim Victor</i>	321
C. Apple MLX: Machine Language Entry Program	
<i>Tim Victor</i>	325
D. Disk Instructions	331
Index	341
Disk Coupon	343

Foreword

High-quality, low-cost software: COMPUTE! Publications' *Third Book of Apple* contains more than two dozen of the best Apple programs published in *COMPUTE!* magazine and *COMPUTE!'s Apple Applications Special*. Chosen for their excellence, all programs are ready to type in and use. Of course, if you want to save yourself typing time, you can order the companion disk—and immediately dive into an arcade game, spreadsheet, or animation maker.

Glance through this book and you'll see what we mean.

"Miami Ice" may have nothing to do with fast cars and fashion in Florida, but you'll enjoy this arcade-style game until you can't turn the steering wheel another inch.

Extraordinary software like "Skyscape," which shows you the night sky from anywhere in the world at anytime, combines education and entertainment in a single package.

SpeedCalc, our all machine language spreadsheet, rivals commercial software in its power and ease of use.

"Apple Animator" puts you in charge of a miniature animation studio, where you create frames, then rapidly flip them.

Transform your computer: an entire chapter is devoted to turning your Apple II-series computer into a Macintosh-like computer—complete with pull-down menus, windows, and mouse movement.

And utilities and programming aids like "MultiMemory" help you do everything from customizing the keyboard to partitioning the computer's memory.

That's just a part of what you'll find inside *COMPUTE!'s Third Book of Apple*. Every article is understandable; every program is ready to use. There are even programs to insure that you type in listings right the first time.

COMPUTE!'s Third Book of Apple—an extraordinary collection of Apple II software.

If you prefer to buy the disk which contains all the programs in this book rather than typing them in, just use the convenient coupon in the back, or call toll-free 1-800-346-6767 (in NY 1-212-887-8525).



1

Recreation



Tug-A-War

Mark Tuttle

Apple version by Tim Victor

"Tug-A-War" isn't as easy at it looks. This two-player strategy game is actually a stiff test of your concentration and ability to think ahead. For all Apple II series computers, using either DOS 3.3 or ProDOS.

Nearly everyone has played tug of war.

The traditional game pits two players or teams at opposite ends of a rope. At the middle of the rope is a flag, and each side tries to pull the flag into its territory. "Tug-A-War" is based on a similar concept. In this version, the flag is replaced with a round ball shape, and each player tries to maneuver the ball onto his or her side of the screen. Like many two-player games, the difficulty of Tug-A-War depends somewhat on the intelligence of your opponent. But even at its simplest level, you'll find that skill and foresight are essential to success.

Battle of the Colors

Type in Tug-A-War, save a copy to disk, and run it.

You'll see two sets of colored boxes, one above the other. The lower, longer series of squares is the playing field. Near the middle of the playfield area is a round ball. The outermost boxes at the ends of the playfield represent each player's home position. Players alternate turns, each trying to move the ball in his or her own direction, until it reaches one of the home squares.

So far, so good—but how do you move the ball? It's done not by pulling a rope, but by changing the colors of boxes in the playfield. The color of the square under the ball determines in which direction it moves and how far it travels. On any given turn, the ball can move either one or two squares to the left, or one or two squares to the right.

Take a look at the top of the screen. Those four boxes show you which colors are linked to which directions. The leftmost box, for instance, shows you which color makes the ball move one square to the left. The next box to the right shows you which color makes it move *two* squares to the left.

The second pair of boxes show you which colors make the ball move in the opposite direction—to the right. By changing the color of the box where the ball is currently located, you can make it move toward your home square.

The playfield contains 11 boxes. When the game begins, each of these boxes is randomly given one of the four colors shown at the top of the screen. On each turn, you may change the color of one, several, or all of the boxes (however, you must always change at least one box). Below each box is a number which represents its distance from the home position of the player whose turn it is. For example, if you're the player on the left, on your turn the boxes are numbered 1, 2, 3, and so on, from left to right (the tenth box is marked with 0, the eleventh with an A). When it's the right player's turn, the numbering is reversed (the rightmost box is 1).

To take a turn, select a number corresponding to the numbers shown below the boxes in the playfield. Just press a single key. Press a number key from 1 through 0 to select one of the first ten values, or press the A key to choose the eleventh box. The number you choose determines how many boxes change color. For instance, if you press 1, only one box (the one nearest your home square) changes color. If you press 2, the two boxes nearest your home box change, and so on.

Where do the new colors come from? Every box cycles through the same series of four colors shown in the uppermost set of boxes, going from left to right. If the colors shown there are white-blue-red-purple (the exact colors may be different on your computer), then a white square always changes to blue, a blue square always changes to red, a purple square changes to white, and so on, down the line. In other words, the box's current color determines which color it becomes after the next color change.

Though every turn involves at least one color change, the ball doesn't necessarily move on every turn. It moves only when you change all the boxes between your home position and the current position of the ball. For example, if the ball is three boxes from your home square, then you must change the color of at least three boxes to move it at all.

Foresight Rewarded

As you can see, there's a lot more to this game than what appears at first glance. You might be tempted to try to move the

ball as often as possible, for instance, even though that's usually a losing strategy. Remember, the direction the ball moves depends on the color of its square *before* you take the turn.

In many cases, you'll want to move the ball only if it's on a color that moves it toward your goal. But like other games of strategy, Tug-A-War rewards the player who looks beyond the current move and tries to set things up for future moves; sometimes it's wise to make a small, temporary sacrifice so that you can reap larger rewards later in the game. Because the boxes change colors in the same sequence, the effect of your own move is always completely predictable. However, since a single turn can change the color of many boxes, dramatic changes of fortune are also possible.

Tug-A-War

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```

3B 100 GOSUB 400
8A 110 HGR : HOME
5C 120 HCOLOR= 3: FOR TD = - 1 TO 1 STEP 2: FOR TN =
      0 TO 1: FOR TX = - 1 TO TN STEP 2: GOSUB 500
      : NEXT : NEXT : NEXT
DC 130 VP = 40
A1 140 FOR I = 0 TO 3: HC = CT(I): PS = 4.5 + I: GOSUB
      430: NEXT
E8 150 VP = 146: HC = 1: PS = 0: GOSUB 430: GOSUB 460
AE 160 FOR I = 0 TO 10: BC(I) = INT ( RND (1) * 4): HC
      = CT(BC(I)): PS = I + 1: GOSUB 430: NEXT
C9 170 HC = 6: PS = 12: GOSUB 430: GOSUB 460
F9 180 BP = 5: GOSUB 470
69 190 VTAB 21: FOR I = 1 TO 11: HTAB I * 3 + 2: IF
      I < 10 THEN PRINT CHR$ (48 + I);
89 200 IF I = 10 THEN PRINT "0";
15 210 IF I = 11 THEN PRINT "A";
E4 220 NEXT : VTAB 23: HTAB 1: PRINT "GREEN'S MOVE:"
      ;
46 230 GOSUB 520: A = A - 1: IF (BP <= A) THEN BP =
      BP + JT(BC(BP))
97 240 FOR I = 0 TO A: BC(I) = BC(I) + 1 - 4 * (BC(I)
      = 3): HC = CT(BC(I)): PS = I + 1: GOSUB 430: N
      EXT
54 250 GOSUB 470
16 260 IF BP < 0 OR BP > 10 THEN 360
67 270 VTAB 21: FOR I = 1 TO 11: HTAB (12 - I) * 3 +
      2: IF I < 10 THEN PRINT CHR$ (48 + I);
99 280 IF I = 10 THEN PRINT "0";
25 290 IF I = 11 THEN PRINT "A";

```

```

80 300 NEXT : VTAB 23: HTAB 1: PRINT "BLUE'S MOVE: "
;
IF 310 GOSUB 520:A = 11 - A: IF (BP >= A) THEN BP =
    BP + JT(BC(BP))
5A 320 FOR I = 10 TO A STEP - 1:BC(I) = BC(I) + 1 -
    4 * (BC(I) = 3):HC = CT(BC(I)):PS = I + 1: GO
    SUB 430: NEXT
51 330 GOSUB 470
13 340 IF BP < 0 OR BP > 10 THEN 360
9F 350 GOTO 190
92 360 PS = 12 * (BP > 0) - 1: HCOLOR= 4 * (BP > 0):
    GOSUB 490
J3 370 VTAB 23: HTAB 1: IF BP < 0 THEN PRINT "GREEN
    WINS ": GOTO 390
6C 380 IF BP > 10 THEN PRINT "BLUE WINS "
0A 390 GET A$: GOTO 110
DE 400 FOR I = 0 TO 3: READ CT(I): NEXT
3B 410 FOR I = 0 TO 3: READ JT(I): NEXT : RETURN
84 420 DATA 3,5,6,2,-1,-2,1,2
8C 430 HCOLOR= HC: FOR YP = VP TO VP + 10
9A 440 HPLLOT PS * 21 + 1,YP TO PS * 21 + 17,YP: NEXT
1E 450 RETURN
AB 460 HCOLOR= 3: FOR YP = VP + 1 TO VP + 9 STEP 2:
    HPLLOT PS * 21 + 1,YP TO PS * 21 + 17,YP: NEXT
    : RETURN
52 470 IF BP < 0 OR BP > 10 THEN RETURN
19 480 HCOLOR= 4 * (CT(BC(BP)) > 3):PS = BP
74 490 FOR YP = VP + 3 TO VP + 7: HPLLOT PS * 21 + 27
    ,YP TO PS * 21 + 32,YP: NEXT : RETURN
0F 500 TP = 124 + (TD + TN) * 21 + TN * TX * 4:TL =
    TP + TD * 3:TR = TP - TD * 3
E5 510 HPLLOT TR,60 TO TL,57 TO TR,54: RETURN
86 520 POKE 49168,0: GET A$: IF A$ = CHR$ (3) THEN E
    ND
89 530 IF A$ = CHR$ (3) THEN END
CB 540 IF A$ < > "A" AND A$ < > "a" AND (A$ < "0" OR
    A$ > "9") THEN 520
36 550 IF A$ = "A" OR A$ = "a" THEN A$ = CHR$ (59)
38 560 IF A$ = "0" THEN A$ = CHR$ (58)
51 570 A = ASC (A$) - 48: RETURN

```

Switchbox

Todd Heimarck

Apple version by Tim Victor

Don't be fooled by "Switchbox." This game of flipping switches and falling balls may look simple, but it takes time to master and permits many variations. Play it on any Apple II series computer with DOS 3.3 or ProDOS.

Playing "Switchbox" is like putting dominos in place for a chain reaction—either you're setting them in position or you're knocking them over. Winning requires skill and a sense of when to go for points and when to lay back and wait for a better board.

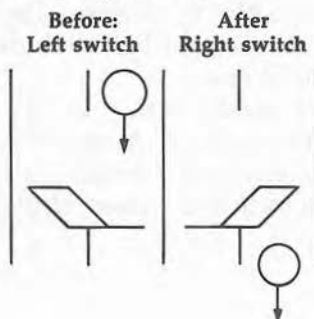
The goal in the game is simple: Try to score more points than your opponent by dropping balls into a boxful of two-way switches. Each switch has a trigger and a platform. If the ball lands on an empty platform, it stops dead. But if it hits a trigger, it reverses the switch and continues. In many cases, dropping a single ball creates a cascading effect—one ball sets another in motion, which sets others in motion, which in turn...you get the idea.

Type in the program and save a copy before you run it.

A Box of Switches

Switchbox is a tale of twos: Each switch has two parts, two positions, two states, two paths in, and two paths out. The two parts are the platform and the trigger. A switch can lean to the left (platform left, trigger right) or to the right (platform right, trigger left):

Figure 1. Trigger States



The trigger is weak and always allows balls to pass. But the platform is strong enough to hold a single ball. So the platform either holds a ball—it's full—or it doesn't and is empty. When a ball sits on a platform, the switch is said to be *loaded*, or *full*.

Figure 2. A Loaded Trigger

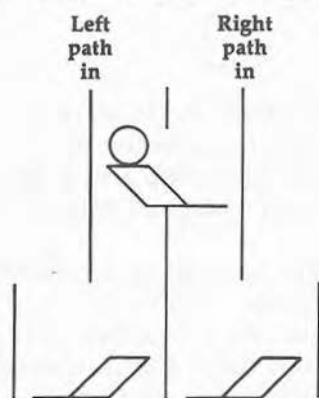


Figure 2 shows a full switch over two empty switches. The platform holds a ball and leans to the left; the trigger extends to the right. Note that the switch on top has two pathways leading in, the left path and the right, and that the right path leading out is the left path into one of the switches below. The left path of the top switch leads into the right path of the other, the switch below and to the left.

If you drop a ball down the righthand path, it hits the trigger and flips that switch to the right. Then it continues down, hits the lefthand trigger below, and flips that switch as well.

In the meantime, the ball on the platform is set in motion (when the switch is flipped) and then hits the trigger. The top switch is reset to point to the left. The second ball then drops a level to the platform below, where it stops.

The playing field is built of five levels, with four switches in the first level and eight in the bottom level. At the beginning of the game, there are no balls on the field—all platforms are empty—and the position of each switch is chosen randomly.

Moving Down the Path

Players alternate dropping balls into one of eight entry points. These balls (and others) may or may not make it all the way through the switchbox to one of the 16 exit paths. Balls fall straight down (with one exception), so their movements are always predictable. When a ball hits an empty switch, one of two things can happen. If it lands on the empty platform, it stops dead in its tracks. But if it lands on a trigger, it falls through to the next level below.

Moving balls always make it through loaded switches. Triggers allow balls to continue and move the switch to the other position. If it's loaded, the dead ball on the platform is put into motion and hits the trigger that just moved over. This makes the switch go back to its original position, but with an empty platform. So when a ball hits the trigger of a loaded switch, its motion continues unabated. The switch moves, the ball on the platform begins to fall, and it hits the newly placed trigger. The newly emptied switch moves back again, and the two balls drop to the next level.

There's one more possibility: a ball dropping onto a platform which already holds a ball. A platform can't hold more than one ball, so when this happens, one of the balls slides over to the trigger. Thus, the ball doesn't move straight down—it slides over to the next pathway. This is the exception to the rule that balls drop in a straight line. Of course, when the ball hits the trigger, the switch changes position, causing the other ball to drop and hit the trigger.

The Chain Reaction

At the game's start, all platforms are empty, so four of eight entry paths are blocked. Remember that your turn ends when a ball hits an empty platform and stops. As the switches fill up, the chances increase that a ball will descend through several levels. The goal is to score points by getting balls to pass all the way through the maze of the switchbox. The best way to collect a lot of points is to cause a chain reaction.

A ball that hits a loaded switch from either side continues on its way. The previously inert ball on the platform starts moving. One enters, two exit. If both of those balls encounter full platforms, four drop from the switches. The pathways are staggered, so the effects can spread outward, with more and more balls cascading toward the bottom.

Rather than taking an easy point or two, it's often worthwhile to build up layers of loaded switches. Watch out for leaving yourself vulnerable, though. Because players take turns, you'll want to leave positions where your opponent's move gives you a chance to create a chain reaction. The best strategy is to play defensively. Look ahead a move or two, and watch for an opening that allows you to score several points at once.

Four Quarters

A game of Switchbox always lasts four rounds. In the first (equality), each exit counts for two points. Your goal is to score ten points. The second quarter has more points available as well as a higher goal. If you look at the exits, you'll see that the farther away from the middle, the higher the point value. The numbers increase in a Fibonacci sequence: 1, 2, 3, 5, 8, and so on. Each number is the sum of the previous two ($1+2$ is 3, $2+3$ is 5, $3+5$ is 8, and so on). The target score in round 2 is 40.

In round 3, the numbers are a bit lower. They increase arithmetically (1, 2, 3, 4, up to 8 in the corners). A goal of 20 points brings you to round 4, where you can score big. Here the numbers are squares: 1, 4, 9, 16, 25, all the way to 64 at the edges. In rounds 2-4, it's sometimes prudent to leave a middle path open for your opponent to score a few points in order to gather a high score on the big numbers to the left and right.

Each round lasts until one player has reached the goal. At that point, the other player has one last turn before the round ends. It's possible to win the round on this last-chance play; watch out for barely topping the goal and leaving a chain reaction open for the other player.

An arrow points to the scoreboard of the current player. On the other side of the screen, you'll see a number where the arrow should be. That's the goal for the current round.

Bonus points are awarded at the conclusion of each round. Four numbers appear below the scorecards. The first is simply the total so far. The second is the total plus a bonus of the goal for the round if the player's points are equal to or greater than the goal. For example, if the goal is 20 and you get 18, there's no bonus. If you score 22, the bonus is the goal for that round (20), and you'd have 42 points. The third number under the scoreboard is the difference between scores for

the rounds. If you win by 2 points, 2 is added to your score (and 2 is subtracted from the other player's score). The final number is the grand total of the first three scores and bonuses. Rounds 1 and 3 are fairly low-scoring with low goals. You may want to seed the field with extra balls during these quarters, so you can collect more points in the second and fourth quarters.

Variations

Although the goal of the game is to score the most points, there's no reason you couldn't agree to play for low score. In a lowball game, you would try to avoid scoring points. You wouldn't necessarily play backwards—you'd have to adjust the strategy of where to place the balls. Fill up the board as much as possible and leave your opponent in a situation where he or she is forced to score points.

The DATA statements at the beginning of the program determine the goal for each round and the point values for the exit paths. You can prolong the game by doubling the goals, which also dilutes the value of a big score at the beginning of a round, preventing one player from winning on the first or second turn. An interesting variation is to assign negative values to some slots. If some paths score negative points, you're forced to think harder about where the balls will drop.

In addition to the numbered keys (1–8), the plus (+) and minus (–) keys are active. Pressing the plus key drops a ball at random down one of the eight entry paths. Pressing the minus key lets you pass your turn to your opponent.

Once you've mastered the regular game, you can add some new rules. Each player gets three passes per half, similar to the three timeouts in a football game. If you don't like the looks of the board, press the minus key to use one of your passes. After one player has skipped a turn, the other player must play (this prevents the possibility of six passes in a row). It's also a good idea to make a rule that a player can't pass on two consecutive turns. You can also give each player two random moves to be played for the opponent. In other words, after making a move, you could inform your opponent that you're going to give him or her one of your random moves, and you would then press the plus key.

Here's one more change you could make: Instead of alternating turns, allow a player to continue after scoring. When a

player drops a ball and scores some points, the other player would have to pass (by pressing the minus key). If the first player scores again, the opponent passes again, and so on, until no more points are scored.

Playing Solitaire

To drop a ball, press a number key (1–8). By using the pass and random-turn options, you can play against the computer. Here are the rules for solitaire play:

- The computer always scores first. At the beginning of every round, the computer plays randomly until at least one point is acquired. Press the plus key for the computer's turn. You must continue passing (skip your turn with the minus key) until the computer puts points on the board.
- After the first score by the computer, you can begin to play. When the computer has a turn, press the plus key for a random move.
- Whenever you score points, you must pass again until the computer scores. When the computer gets more points, you can begin to play again. This rule means you should hold back on the easy scores of a few points; wait until there's an avalanche available.
- If you're the first to reach the goal, the computer gets a last chance. Don't make this move randomly; figure out the best opportunity for scoring and play that move for the last-chance turn.

In the interest of keeping the programs to a manageable length, no attempt was made to provide an "intelligent" computer opponent. Once you become familiar with the game, you might find it an interesting project to try adding some routines that give the computer a rational basis for picking one move over another.

Switchbox

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
4D 100 DIM SW(4,7,1),SP$(1),LB(32,4),AR$(1),PT(4,16)
    ,SC(1,8)
D4 110 SP$(0) = "%&":SP$(1) = "!" + CHR$(34):AR$(
    0) = "<--":AR$(1) = "-->":QR = 1
D3 120 FOR J = 1 TO 4: READ PT(J,0)
6C 130 FOR K = 1 TO 8: READ L:PT(J,K + 8) = L:PT(J,9
    - K) = L: NEXT K,J
```



```

84 140 DATA 10,2,2,2,2,2,2,2,2
AE 150 DATA 40,1,2,3,5,8,13,21,34
85 160 DATA 20,2,3,4,5,6,7,8,9
6E 170 DATA 80,1,4,9,16,25,36,49,64
16 180 HOME : FLASH : PRINT "READING DATA STATEMENTS
- ONE MOMENT": NORMAL
72 190 IF PEEK (768) < > 169 THEN POKE 230,64: GOSUB
970
68 195 IF PEEK (190 * 256) = 76 THEN PRINT CHR$ (4) "
PR#A$35C": GOTO 210
5C 200 POKE 54,92: POKE 55,3: CALL 1002: POKE 6,0
E1 210 TEXT : HOME : INPUT "PLAYER 1: ";P1$: INPUT "
PLAYER 2: ";P2$
32 220 PRINT "IS THIS CORRECT?": GET A$:A = ASC (A$)
: IF A < > 89 AND A < > 121 THEN 210
F6 230 POKE 7,138: POKE 6,0: GOSUB 350: GOSUB 530
0B 240 FOR RR = 1 TO 4: POKE 7,138: VTAB RR + 1: HTA
B 8: PRINT "+": HTAB 30: PRINT "+";
58 250 GOSUB 490
2D 260 POKE 7,141:QR = 1 - QR: VTAB 1: HTAB 28 - QR
* 21: PRINT " ";PT(RR,0);" ";: POKE 7,138: HT
AB 8 + QR * 20: PRINT AR$(QR)
94 270 GOSUB 630: IF SC(1 - QR,RR) > = PT(RR,0) THEN
290
22 280 GOTO 260
69 290 POKE 7,141: FOR J = 0 TO 1: FOR K = 5 TO 8:SC
(J,K) = 0: NEXT K,J
56 300 FOR J = 0 TO 1: FOR K = 1 TO 4:GL = PT(K,0):A
C = SC(J,K):SC(J,5) = SC(J,5) + AC:SC(J,6) =
SC(J,6) + (AC > = GL) * GL:SC(J,7) = SC(J,7)
+ (SC(J,K) - SC(1 - J,K)): NEXT K,J
4E 310 FOR J = 0 TO 1: FOR K = 6 TO 7:SC(J,K) = SC(J
,K) + SC(J,5): NEXT K,J
4B 320 FOR J = 0 TO 1: FOR K = 5 TO 7:SC(J,8) = SC(J
,8) + SC(J,K): NEXT K,J
4A 330 FOR J = 0 TO 1: FOR K = 5 TO 8:Y$ = STR$ (SC(
J,K)):L = LEN (Y$):TX = 6 + J * 31 - L:TY = 3
+ K: VTAB TY: HTAB TX: PRINT Y$: NEXT K,J
3F 340 NEXT RR
03 341 VTAB 15: HTAB 14: PRINT "GAME OVER"
08 342 VTAB 17: HTAB 08: PRINT "PRESS Y TO PLAY AGAI
N"
44 345 GET G$: IF G$ = "" THEN 345
37 346 IF G$ < > "Y" THEN TEXT : HOME : PRINT "SWITC
HBOX OFF": END
CD 348 RUN
E6 350 HOME : HGR2
CE 360 FOR I = 0 TO 5: FOR J = 0 TO 1
8A 370 HTAB 11 - I * 2: VTAB I * 4 + J + 2: PRINT "*"
";: HTAB 27 + I * 2: PRINT "*": NEXT

```

```

2C 380 NEXT : FOR I = 0 TO 4
84 390 HTAB 10 - I * 2: VTAB I * 4 + 4: PRINT CHR$ (
33); CHR$ (34);
97 400 HTAB 27 + I * 2: PRINT CHR$ (37); CHR$ (38);
86 410 HTAB 9 - I * 2: VTAB I * 4 + 5: PRINT CHR$ (3
5); CHR$ (36);
C6 420 HTAB 28 + I * 2: PRINT CHR$ (39); CHR$ (40)
02 430 NEXT
19 440 HCOLOR= 5: FOR I = 0 TO 6: FOR HP = 87 - I *
14 TO 171 + I * 14 STEP 28
7E 450 VS = I * 32 - 28: VE = VS + 50: IF VS < 8 THEN
VS = 8
3D 460 IF VE > 182 THEN VE = 182
22 470 HPLOT HP, VS TO HP, VE: NEXT
D0 480 NEXT : RETURN
CB 490 POKE 7, 141: FOR J = 1 TO 16: K = PT(RR, J): JJ =
2 + J * 2
4D 500 IF K > 9 THEN L = INT (K / 10): L$ = STR$ (L):
GOTO 520
3E 510 L$ = " "
50 520 VTAB 23: HTAB JJ: PRINT L$: HTAB JJ: PRINT RI
GHT$ ( STR$ (K), 1);: NEXT J: RETURN
96 530 FOR J = 0 TO 4: SY = 5 + J * 4: FOR K = 0 TO J
+ 3: SX = 12 - J * 2 + K * 4
80 540 WP = INT ( RND (1) * 2)
15 550 SW(J, K, 0) = WP: SW(J, K, 1) = 0: GOSUB 620
31 560 NEXT K, J
A2 570 POKE 7, 141
BB 580 VTAB 1: HTAB 12: FOR J = 1 TO 8: PRINT J; " ";
: NEXT
E0 590 VTAB 1: HTAB 3 - ( LEN (P1$) = 5): PRINT P1$;
FD 600 HTAB 34 - ( LEN (P2$) = 5): PRINT P2$;
18 610 RETURN
27 620 VTAB SY: HTAB SX: PRINT SP$(WP): RETURN
CB 630 FOR J = 0 TO 32: LB(J, 0) = 0: NEXT : NB = 1
4B 640 GET A$: IF A$ = "-" THEN RETURN
FF 650 IF A$ = "+" THEN A$ = STR$ ( INT ( RND (1) *
8 + 1))
2F 660 A = VAL (A$): IF A < 1 OR A > 8 THEN 640
F7 670 LB(0, 0) = 1: FOR J = 1 TO 3: LB(0, J) = 0: NEXT
: LB(0, 4) = 10 + A * 2
CA 680 EX = 1
86 690 FOR J = 0 TO 32: IF LB(J, 0) THEN EX = 0: GOSU
B 720
2C 700 NEXT : IF EX = 0 THEN 680
19 710 RETURN
F3 720 DY = LB(J, 0): DX = LB(J, 1): LY = LB(J, 2): NY = L
B(J, 3): NX = LB(J, 4): IF (LY + NY) THEN GOSUB
1060
90 730 LB(J, 3) = NY + 1 - (NY = 3) * 4: ON NY + 1 GO
TO 740, 760, 790, 800

```

```

E1 740 IF LY > 4 THEN LB(J,0) = 0: GOTO 840
06 750 GOSUB 1080: ON INT ( RND (1) * 3 + 1) GOTO 88
0,890,900
E4 760 VX = 0: GOSUB 830: IF SW(WY,WX,1) AND (SW(WY,
WX,0) = SD) THEN VX = 1 - 2 * SD:LB(J,1) = VX
:LB(J,3) = NY + 1:BX = NX + VX:LB(J,4) = BX:BY
Y = NY + LY * 4 + 3: GOSUB 1090: GOTO 930
7D 770 IF SW(WY,WX,0) = SD THEN LB(J,0) = 0:SW(WY,WX
,1) = 1: GOSUB 1080: GOTO 920
F1 780 LB(J,3) = NY + 1: GOSUB 1080: ON INT ( RND (1
) * 3 + 1) GOTO 880,890,900
64 790 LB(J,1) = 0:BX = NX + DX:LB(J,4) = BX:BY = NY
+ LY * 4 + 3: GOSUB 1090: GOTO 940
B4 800 LB(J,2) = LY + 1: GOSUB 1080: GOSUB 830:SW(WY
,WX,0) = 1 - SW(WY,WX,0)
62 810 IF SW(WY,WX,1) THEN LB(NB,0) = 1:LB(NB,1) = 0
:LB(NB,2) = LY:LB(NB,3) = 0:LB(NB,4) = NX + 2
- SD * 4:NB = NB + 1:SW(WY,WX,1) = 0:BX = NX
+ 2 - SD * 4:BY = NY + LY * 4 + 1: GOSUB 107
0: GOSUB 950
E9 820 SX = 12 - WY * 2 + WX * 4:SY = 5 + WY * 4:WP
= SW(WY,WX,0): GOSUB 620: GOTO 930
5D 830 WY = LY:JX = (NX / 2) + LY - 6:WX = INT (JX /
2):SD = JX - INT (JX / 2) * 2: RETURN
E0 840 POKE 7,141:SF = PT(RR,NX / 2 - 1)
B6 850 SG = SC(QR,RR) + SF
4F 860 TX = 6 + 31 * QR - LEN ( STR$ (SG))
DC 870 VTAB RR + 1: HTAB TX: PRINT SG:SC(QR,RR) = SG
: POKE 7,138: GOTO 960
05 880 POKE 776,80: GOTO 910
6A 890 POKE 776,160: GOTO 910
21 900 POKE 776,201: GOTO 910
69 910 POKE 781,200: POKE 841,1: POKE 849,196: POKE
798,96: CALL 768: RETURN
44 920 POKE 776,208: POKE 781,220: POKE 841,5: POKE
849,4: POKE 798,97: CALL 768: RETURN
03 930 POKE 776,232: POKE 781,255: POKE 841,0: POKE
849,0: POKE 798,240: CALL 768: RETURN
FA 940 POKE 776,216: POKE 781,240: POKE 841,4: POKE
849,4: POKE 798,240: CALL 768: RETURN
F8 950 POKE 776,160: POKE 781,160: POKE 841,1: POKE
849,96: POKE 798,240: CALL 768: RETURN
EB 960 POKE 776,160: POKE 781,220: POKE 841,6: POKE
849,6: POKE 798,97: CALL 768: RETURN
88 970 FOR I = 768 TO 947: READ A: POKE I,A: NEXT
AF 980 FOR I = 24576 TO 24831: POKE I,128: NEXT
78 990 FOR I = 24832 TO 25087 STEP 4: POKE I,128: PO
KE I + 1,136: POKE I + 2,170: POKE I + 3,136:
NEXT

```



```

B9 1000 FOR I = 35328 TO 35439: READ A: POKE I,A: NE
    XT
51 1010 FOR I = 35552 TO 35559: READ A: POKE I,A: NE
    XT
F5 1020 FOR I = 35568 TO 35575: READ A: POKE I,A: NE
    XT
C0 1030 FOR I = 35704 TO 35711: READ A: POKE I,A: NE
    XT
20 1035 FOR I = 36096 TO 36103: POKE I,0: NEXT
12 1040 FOR I = 36200 TO 36311: READ A: POKE I,A: NE
    XT
D0 1050 FOR I = 36360 TO 36599: READ A: POKE I,A: NE
    XT : RETURN
04 1060 BY = NY + LY * 4 + 2:BX = NX
6B 1070 VTAB BY: HTAB BX: PRINT " ": RETURN
C5 1080 BX = NX:BY = NY + LY * 4 + 3
F8 1090 VTAB BY: HTAB BX: PRINT "0": RETURN
AF 1100 DATA 169,1,141,88,3,160,0,169
5B 1110 DATA 160,141,49,3,169,255,141,59
5B 1120 DATA 3,173,59,3,141,90,3,78
5A 1130 DATA 88,3,144,12,185,0,145,200
64 1140 DATA 141,89,3,169,128,141,88,3
FC 1150 DATA 78,89,3,144,3,173,48,192
19 1160 DATA 162,0,232,208,253,144,3,173
A3 1170 DATA 48,192,162,159,232,208,253,238
25 1180 DATA 90,3,208,211,24,173,59,3
BC 1190 DATA 233,3,141,59,3,173,49,3
51 1200 DATA 105,3,141,49,3,144,186,96
6C 1210 DATA 8,5,0,255,216,120,133,69
25 1220 DATA 134,70,132,71,166,7,10,10
AD 1230 DATA 176,4,16,62,48,4,16,1
44 1240 DATA 232,232,10,134,27,24,101,6
50 1250 DATA 133,26,144,2,230,27,165,40
F2 1260 DATA 133,8,165,41,41,3,5,230
CC 1270 DATA 133,9,162,8,160,0,177,26
95 1280 DATA 36,50,48,2,73,127,164,36
39 1290 DATA 145,8,230,26,208,2,230,27
B7 1300 DATA 165,9,24,105,4,133,9,202
5B 1310 DATA 208,226,165,69,166,70,164,71
64 1320 DATA 88,76,240,253
5C 1330 DATA 0,0,0,0,0,0,0,0
90 1340 DATA 0,64,96,112,120,124,126,127
E0 1350 DATA 127,63,31,15,7,3,1,0
E3 1360 DATA 64,96,112,120,124,126,127,127
7D 1370 DATA 63,31,15,7,3,1,0,0
3F 1380 DATA 127,126,124,120,112,96,64,0
A9 1390 DATA 0,1,3,7,15,31,63,127
81 1400 DATA 126,124,120,112,96,64,0,0
44 1410 DATA 1,3,7,15,31,63,127,127
AB 1420 DATA 0,0,0,0,0,0,0,127
F7 1430 DATA 127,127,127,127,127,127,127,127

```

EE 1440 DATA 0,0,28,62,127,62,28,0
 66 1450 DATA 0,0,0,0,0,0,0,0
 7C 1460 DATA 0,0,0,127,127,0,0,0
 7C 1470 DATA 0,12,6,127,127,6,12,0
 0B 1480 DATA 0,24,48,127,127,48,24,0
 9A 1490 DATA 0,0,28,62,62,62,28,0
 0D 1500 DATA 0,0,0,0,14,0,0,0
 FF 1510 DATA 0,0,0,14,0,14,0,0
 F1 1520 DATA 0,60,102,48,24,0,24,0
 9C 1530 DATA 0,60,102,118,110,102,60,0
 C9 1540 DATA 0,24,28,24,24,24,60,0
 B4 1550 DATA 0,60,102,48,12,102,126,0
 29 1560 DATA 0,60,102,48,96,102,60,0
 B2 1570 DATA 0,48,56,52,126,48,48,0
 3B 1580 DATA 0,126,6,62,96,102,60,0
 E1 1590 DATA 0,60,6,62,102,102,60,0
 A0 1600 DATA 0,126,96,48,24,12,12,0
 B5 1610 DATA 0,60,102,60,102,102,60,0
 0F 1620 DATA 0,60,102,102,124,48,24,0
 B6 1630 DATA 0,24,48,126,126,48,24,0
 AD 1640 DATA 0,124,102,102,126,102,102,0
 8D 1650 DATA 0,62,102,102,62,102,126,0
 69 1660 DATA 0,60,102,6,6,102,62,0
 CB 1670 DATA 0,62,102,102,102,102,62,0
 0C 1680 DATA 0,126,6,6,62,6,126,0
 48 1690 DATA 0,126,6,6,62,6,6,0
 F0 1700 DATA 0,60,102,6,118,102,62,0
 73 1710 DATA 0,102,102,102,126,102,102,0
 CC 1720 DATA 0,24,24,24,24,24,24,0
 77 1730 DATA 0,96,96,96,96,102,60,0
 64 1740 DATA 0,102,102,54,30,102,102,0
 DD 1750 DATA 0,6,6,6,6,6,126,0
 87 1760 DATA 0,102,126,102,102,102,102,0
 0D 1770 DATA 0,62,102,102,102,102,102,0
 41 1780 DATA 0,60,102,102,102,102,60,0
 91 1790 DATA 0,62,102,102,62,6,6,0
 9C 1800 DATA 0,60,102,102,102,54,108,0
 40 1810 DATA 0,62,102,102,62,102,102,0
 11 1820 DATA 0,60,102,12,48,102,62,0
 70 1830 DATA 0,126,24,24,24,24,24,0
 FD 1840 DATA 0,102,102,102,102,102,62,0
 02 1850 DATA 0,102,102,102,102,102,24,0
 D4 1860 DATA 0,102,102,102,102,126,102,0
 1A 1870 DATA 0,102,102,102,60,102,102,0
 B0 1880 DATA 0,102,102,102,60,24,24,0
 CE 1890 DATA 0,126,48,24,12,6,126,0
 5C 1900 DATA 0,0,0,0,0,0,0,0
 60 1910 DATA 0,0,0,0,0,0,0,0
 64 1920 DATA 0,0,0,0,0,0,0,0
 6A 1930 DATA 0,0,24,60,60,24,0,0

High Rise

Charles McGuyer

Apple version by Tim Victor

You're a construction worker, trapped in a partially completed high-rise building after dark. Can you make it safely to the ground floor without being zapped by the patrol robot? This unique game, written entirely in machine language, is one of the best Apple arcade-style games we've ever published. Works with DOS 3.3 or ProDOS.

The time is the not-too-distant future. The place is a downtown high-rise building under construction. You're just finishing the day's work when you realize it's already dark. Everyone else has gone home, leaving you alone in a shadowy, multistory maze of naked girders and bare concrete. A chill creeps down your spine as you think about the recently installed antitheft robot. It patrols the structure from dusk to dawn, automatically disposing of any intruder.

Your only hope is to use the temporary elevators. They move randomly during the night hours, going up and down, stopping at some floors, skipping others. With skill and a little luck, you just might evade the dangers around you and make it safely to the ground floor—but it won't be a cakewalk.

Type Type

"High Rise" must be entered with the "Apple MLX" machine language entry program found in Appendix C. Since High Rise loads into the memory area normally used by BASIC programs, you must relocate the start of BASIC memory *before* loading MLX to type High Rise. To do this, enter the following line in direct mode (without a line number) and press Return:

POKE 104,28: POKE 7168,0: NEW

Next, load and run MLX. Follow the MLX instructions carefully, using these addresses:

Starting address: 0801

Ending address: 1BD8

After you finish typing High Rise, save at least one copy on disk. Once that's done, you can activate High Rise by typing **BRUN filename** (substituting your own filename, of course).

Run Away

The object of High Rise is to make your way to the ground floor via the elevators while evading the patrol robot. When the game begins, you'll see several floors of the building and a number of elevators moving up and down. Move your player with keyboard controls: Press the left-arrow key to move left, the right-arrow key to move right, and the space bar to stop.

You can jump on any elevator that comes to your floor (move into the elevator and it picks you up automatically), but there's no way to control its direction or how far it goes.

These are just temporary elevators, used to transport materials and workers during daytime hours. The trick is to catch one that's moving in the direction you want and get off to catch another before it starts moving in an unwanted direction.

When you reach the lowest floor shown, the screen scrolls up one floor, revealing the next lower level. Once you reach ground floor, the player sprints off the screen to safety and you can play another game.

The patrol robot always starts on an upper floor and moves systematically through the building, traveling up and down through special shafts that are closed to you. Designed to discourage theft and vandalism, its technique is simple and effective: It pushes any intruders (including you) off the building. If it runs into an elevator and detects you inside, it sends a high-voltage charge through the elevator shell until you drop.

Note: When boarding an elevator, make sure that your player is positioned exactly in the center of the elevator shaft (the double-dotted lines). If you're just a bit to one side, the elevator will not pick you up.

When the game begins, you have three players. Each time you're zapped or fall from the building, you lose a player. Play ends when all three have been lost. When the game starts, you're on the twelfth floor. Moving down a level earns you 100 points. If you reach the bottom safely, you'll have another chance to play, beginning at a higher floor. High Rise keeps track of the highest score attained in the current session as well as your score in the current game.

High Rise

For mistake-proof program entry, use "Apple MLX" (Appendix C) to type in this program.

START ADDRESS: 0801

END ADDRESS: 1BD8

```
0801: A9 00 85 EC A9 80 85 ED AD
0809: 20 DC 13 20 A4 1B 20 6F 06
0811: 17 A9 20 85 E6 8D E0 1B BD
0819: A9 00 8D D7 1B 8D D6 1B 05
0821: A9 28 8D DD 1B A9 C0 8D 2E
0829: DE 1B 20 C7 15 A9 40 85 45
0831: E6 20 C7 15 A9 00 8D FE 6E
0839: 1B 8D FF 1B 2C 57 C0 2C 58
0841: 52 C0 2C 54 C0 2C 50 C0 8D
0849: A9 70 8D F8 1B A9 0E 8D B4
0851: F9 1B A9 C4 8D FA 1B A9 DE
0859: 0E 8D FB 1B A9 00 8D 74 E1
0861: 0E 8D 81 0E 8D 8E 0E 8D 3D
0869: 9B 0E 8D A4 0E 8D B1 0E DE
0871: 8D BB 0E 8D C8 0E 8D F6 62
0879: 1B A9 0A 8D F4 1B A9 00 03
0881: 8D F3 1B A9 05 8D 9E 0E FC
0889: A9 02 8D 84 0E 8D 85 0E A8
0891: 8D 86 0E AD F6 1B F0 03 AF
0899: 4C 10 0A A9 01 8D E3 1B D0
08A1: A9 01 8D E2 1B A9 00 8D B3
08A9: E4 1B A9 56 8D E5 1B A9 71
08B1: 00 8D E6 1B A9 00 8D E7 04
08B9: 1B A9 01 8D EC 1B A9 00 E1
08C1: 8D ED 1B 8D EA 1B A9 13 7A
08C9: 8D EE 1B A9 00 8D F1 1B 8F
08D1: A9 06 8D EF 1B A9 00 8D F5
08D9: F0 1B AD F3 1B AC F4 1B AE
08E1: 8C F3 1B 6A 90 0D EE F3 C9
08E9: 1B A9 7C 8D E5 1B A9 39 82
08F1: 8D EE 1B A9 FF 8D E9 1B A7
08F9: A9 00 8D F7 1B 20 A4 14 C6
0901: 20 3C 0D 20 45 0C 20 81 F1
0909: 0D 20 61 1A 20 5A 17 20 30
0911: 33 09 20 C0 09 20 A4 14 35
0919: 20 6D 15 20 B1 19 20 A7 15
0921: 0B 20 09 0B 20 BD 0A 20 BE
0929: 7B 0A AD F7 1B F0 CE 4C 37
0931: 94 08 AD E5 1B A0 00 C9 C8
0939: A2 B0 06 69 26 C8 4C 38 45
0941: 09 98 18 6D F3 1B CD E8 68
0949: 1B F0 73 8D E8 1B A2 02 67
0951: 8E B4 0E C9 0A 90 08 EE C7
0959: B4 0E E9 0A 4C 54 09 69 56
```

0961: 02 8D B5 0E A9 00 8D B1 89
 0969: 0E 98 D0 52 A2 02 FE 84 87
 0971: 0E BD 84 0E C9 0C D0 08 93
 0979: A9 02 9D 84 0E CA 10 EE 87
 0981: A9 00 8D 81 0E 20 35 1B A8
 0989: AD F3 1B D0 2B 20 05 0E D1
 0991: AD F4 1B 18 69 04 8D F4 08
 0999: 1B EE CC 0E AD CC 0E C9 F5
 09A1: 0C D0 08 A9 02 8D CC 0E 77
 09A9: EE CB 0E A9 00 8D C8 0E 5B
 09B1: A9 01 8D F7 1B 4C 01 0A 20
 09B9: CE F3 1B 20 EA 15 60 AD AF
 09C1: E5 1B 38 ED EE 1B C9 08 F2
 09C9: 10 35 C9 EE 30 31 AD E6 E1
 09D1: 1B 38 ED EF 1B C9 02 10 50
 09D9: 26 C9 FF 30 22 20 47 1B AF
 09E1: 20 CA 0D A9 01 8D F7 1B 3C
 09E9: CE 9E 0E A9 00 8D 9B 0E E2
 09F1: A9 02 CD 9E 0E D0 05 A9 64
 09F9: 01 8D F6 1B 20 01 0A 60 F9
 0A01: A9 18 8D 0F 0A 20 60 0A 2E
 0A09: CE 0F 0A D0 F8 60 00 A9 89
 0A11: 0A A0 49 20 FC 1A A9 80 A1
 0A19: 8D 4D 0A 20 60 0A 2C 10 1E
 0A21: C0 AD 00 C0 10 FB 2C 10 E5
 0A29: C0 C9 CE F0 12 C9 EE F0 7F
 0A31: 0E A9 40 8D 4D 0A 20 60 CA
 0A39: 0A 20 60 0A 4C 0F 08 2C E1
 0A41: 54 C0 2C 51 C0 4C D0 03 26
 0A49: 20 2E 20 53 2E 34 07 1B 9D
 0A51: 1D 10 1E 1E 01 19 01 1F 2B
 0A59: 1A 01 1C 20 14 1F 00 20 7D
 0A61: A4 14 20 6D 15 20 B1 19 4D
 0A69: 20 A7 0B 20 A4 14 20 3C CC
 0A71: 0D 20 81 0D 20 61 1A 4C 1C
 0A79: 5A 17 AD E2 1B F0 36 10 7D
 0A81: 1B AD E6 1B D0 07 AD E7 03
 0A89: 1B C9 02 90 29 AD E7 1B D1
 0A91: 38 E9 02 B0 1D 69 07 CE F2
 0A99: E6 1B 10 16 AD E6 1B C9 54
 0AA1: 1C B0 13 AD E7 1B 18 69 72
 0AA9: 02 C9 07 90 05 EE E6 1B E7
 0AB1: E9 07 8D E7 1B 60 A9 00 5A
 0AB9: 8D E2 1B 60 2C E9 1B 30 26
 0AC1: 0D AD E5 1B 18 69 1C 38 0D
 0AC9: E9 26 90 2F D0 FA AD 00 2F
 0AD1: C0 10 28 8D 10 C0 C9 8D CC
 0AD9: F0 22 C9 A0 F0 13 C9 8B 22
 0AE1: F0 07 C9 95 D0 15 A9 01 F1
 0AE9: 2C A9 FF A0 FF 8C E9 1B A9

```

0AF1: 2C A9 00 8D E2 1B A9 01 37
0AF9: 8D E3 1B 60 2C 10 C0 AD 08
0B01: 00 C0 10 FB 2C 10 C0 60 8C
0B09: AD EA 1B F0 03 4C 84 0B 80
0B11: AD E5 1B 18 69 09 CD EE 56
0B19: 1B D0 0A AD E6 1B CD EF 3C
0B21: 1B B0 38 90 39 90 03 A9 BC
0B29: 02 2C A9 FE 8D EB 1B AD 70
0B31: F1 1B 4A A9 0B 2C EB 1B E6
0B39: 10 04 90 06 B0 02 B0 02 BB
0B41: A9 F5 1B 69 0E CD EF 1B E5
0B49: D0 0F A9 03 CD F0 1B D0 2A
0B51: 08 AD EB 1B 8D EA 1B D0 25
0B59: 2A 90 03 A9 02 2C A9 FE B6
0B61: 8D EB 1B 18 6D F0 1B C9 4D
0B69: 07 90 14 2C EB 1B 10 09 61
0B71: 1B 69 07 CE EF 1B 4C 80 C0
0B79: 0B 38 E9 07 EE EF 1B 8D CB
0B81: F0 1B 60 18 6D EE 1B 8D 4F
0B89: EE 1B 18 69 13 38 E9 26 EA
0B91: 90 13 D0 FA A9 00 8D EA D1
0B99: 1B 2C EB 1B 10 04 CE F1 97
0BA1: 1B 60 EE F1 1B 60 A0 00 F5
0BA9: 8C FC 1B B9 C5 0D 30 09 10
0BB1: AA CA 8A 99 C5 0D 4C 39 EE
0BB9: 0C 18 B9 B1 0D 79 C0 0D 0B
0BC1: 99 B1 0D C9 06 F0 04 C9 15
0BC9: 9E D0 09 38 A9 00 F9 C0 0A
0BD1: 0D 99 C0 0D CC E9 1B D0 D2
0BD9: 0B B9 B1 0D 18 69 04 8D E6
0BE1: E5 1B D0 33 B9 B1 0D 18 C5
0BE9: 69 06 CD E5 1B 90 28 E9 A3
0BF1: 05 CD E5 1B B0 21 B9 B6 A0
0BF9: 0D CD E6 1B D0 19 B9 BB B2
0C01: 0D CD E7 1B D0 11 A9 00 DF
0C09: 8D E2 1B 8C E9 1B B9 B1 AD
0C11: 0D 18 69 04 8D E5 1B B9 17
0C19: B1 0D C9 9E F0 12 38 E9 9A
0C21: 06 F0 0D E9 26 90 11 D0 1F
0C29: FA 20 C6 18 29 E0 D0 08 97
0C31: 20 C6 18 4A 4A 99 C5 0D 04
0C39: C8 8C FC 1B C0 05 F0 03 29
0C41: 4C AC 0B 60 CE E3 1B D0 1F
0C49: 1C AD E2 1B F0 03 A9 02 D1
0C51: 2C A9 30 8D E3 1B EE E4 17
0C59: 1B AD E4 1B C9 04 D0 05 BD
0C61: A9 00 8D E4 1B AD E4 1B C2
0C69: 0A 0A 0A AE E2 1B F0 1C B6
0C71: 30 0D 69 FC 85 FA A9 00 4D
0C79: 69 0C 85 FB 4C 97 0C 69 FB

```



```

0C81: 1C 85 FA A9 00 69 0D 85 48
0C89: FB 4C 97 0C 69 DC 85 FA 2B
0C91: A9 00 69 0C 85 FB A0 00 C9
0C99: 8C FC 1B AD FC 1B 0A 0A E7
0CA1: A8 B1 FA 85 EE C8 B1 FA 2B
0CA9: 85 EF C8 18 AD E5 1B 71 C7
0CB1: FA 8D D7 1B C8 18 AD E6 40
0CB9: 1B 8D D6 1B AD E7 1B 71 04
0CC1: FA C9 07 90 05 EE D6 1B 60
0CC9: E9 07 8D D5 1B 20 C0 14 96
0CD1: EE FC 1B AD FC 1B C9 02 C8
0CD9: 90 C1 60 74 12 01 01 92 27
0CE1: 12 09 01 B0 12 01 01 92 99
0CE9: 12 09 01 74 12 01 01 92 DD
0CF1: 12 09 01 B0 12 01 01 92 A9
0CF9: 12 09 01 74 12 02 01 CE 2E
0D01: 12 0A 00 74 12 01 01 EC 71
0D09: 12 09 04 74 12 02 01 0A DA
0D11: 13 0A 00 74 12 01 01 EC 02
0D19: 12 09 04 B0 12 02 01 28 CC
0D21: 13 0A 05 B0 12 01 01 46 CF
0D29: 13 09 05 B0 12 02 01 64 B9
0D31: 13 0A 04 B0 12 01 01 46 BF
0D39: 13 09 05 CE EC 1B D0 0D 2F
0D41: A9 14 8D EC 1B A9 01 4D 84
0D49: ED 1B 8D ED 1B AD ED 1B 38
0D51: F0 02 A9 1E 18 69 82 85 6C
0D59: EE A9 13 85 EF 90 02 E6 BC
0D61: EF AD EF 1B 8D D6 1B AD 3A
0D69: F0 1B 8D D5 1B AD EE 1B 5A
0D71: 38 ED ED 1B ED ED 1B 69 59
0D79: 01 8D D7 1B 20 C0 14 60 B0
0D81: A9 00 8D FC 1B A9 BE 85 74
0D89: EE A9 13 85 EF AC FC 1B 87
0D91: C0 05 F0 1B B9 B1 0D 8D 59
0D99: D7 1B B9 B6 0D 8D D6 1B 70
0DA1: B9 BB 0D 8D D5 1B 20 C0 1E
0DA9: 14 EE FC 1B 4C 8E 0D 60 F1
0DB1: 10 30 50 70 90 00 07 0E 91
0DB9: 14 1B 04 01 00 06 01 FE 4E
0DC1: 02 FE 02 FE FF FF FF FF CC
0DC9: FF 20 A4 14 20 6D 15 20 C2
0DD1: B1 19 A9 FF 8D E9 1B AD 38
0DD9: E5 1B 18 69 08 8D E5 1B A4
0DE1: 20 A7 0B 20 A4 14 20 3C 4B
0DE9: 0D AD E5 1B C9 A8 B0 03 B9
0DF1: 20 45 0C 20 81 0D 20 61 D2
0DF9: 1A 20 5A 17 AD E5 1B C9 EA
0E01: A8 90 C6 60 A9 FF 8D E2 BF
0E09: 1B A9 01 8D E3 1B A0 59 3C

```

```

0E11: A9 0E 20 FC 1A A9 80 8D 5F
0E19: 5D 0E 20 A4 14 20 6D 15 C6
0E21: 20 B1 19 20 A7 0B A9 FF 9B
0E29: 8D E9 1B 20 7B 0A 20 A4 D4
0E31: 14 20 3C 0D AD E2 1B F0 D7
0E39: 03 20 45 0C 20 81 0D 20 89
0E41: 61 1A 20 5A 17 AD E2 1B 8E
0E49: D0 D0 A9 0A 8D F3 1B AD F7
0E51: 5D 0E 09 40 8D 5D 0E 60 23
0E59: 4F 7F 55 45 20 34 07 1D F8
0E61: 1A 20 19 0F 01 0E 1A 1B 33
0E69: 1B 17 10 1F 10 0F 00 7D 07
0E71: 0E F8 1B 00 0E 20 1E 0E 71
0E79: 1A 1D 10 00 8A 0E 70 0E 67
0E81: 00 1A 20 02 02 02 02 02 66
0E89: 00 97 0E 7D 0E 00 3C 20 2E
0E91: 17 14 21 10 1E 00 A0 0E A3
0E99: 8A 0E 00 48 22 02 00 AD C9
0EA1: 0E 97 0E 00 6A 20 11 17 79
0EA9: 1A 1A 1D 00 B7 0E A0 0E 42
0EB1: 00 76 22 02 03 00 C4 0E 7F
0EB9: AD 0E 00 98 20 17 10 21 58
0EC1: 10 17 00 00 00 B7 0E 00 A6
0EC9: A4 22 02 03 00 A0 B1 80 37
0ED1: B0 83 80 B0 8D 80 E0 83 F5
0ED9: 80 B0 81 80 F0 83 80 E0 12
0EE1: 80 80 C0 81 80 A0 81 80 98
0EE9: B0 83 80 AC 83 80 F0 81 9B
0EF1: 80 A0 83 80 F0 83 80 C0 46
0EF9: 81 80 E0 80 80 9C 8E 80 2F
0F01: BE 9F 80 F6 9B 80 F6 9B 4E
0F09: 80 E6 99 80 8C 8C 80 E0 D4
0F11: 81 80 F0 83 80 B0 83 80 B4
0F19: B0 83 80 B0 83 80 B0 83 8E
0F21: 80 B0 83 80 B0 83 80 90 49
0F29: 82 80 B0 80 80 D8 81 80 B1
0F31: D8 83 80 B0 83 80 C0 82 D9
0F39: 80 F0 83 80 F0 81 80 E8 C3
0F41: 81 80 D8 83 80 9C 87 80 99
0F49: 8E 87 80 C4 83 80 98 81 BD
0F51: 80 80 86 80 E0 80 80 F0 A3
0F59: 81 80 D0 83 80 B0 83 80 F8
0F61: C0 84 80 F0 83 80 F0 81 A1
0F69: 80 F8 81 80 F8 82 80 9C AD
0F71: 87 80 8E 87 80 C4 83 80 5C
0F79: 98 81 80 80 86 80 B0 80 74
0F81: 80 F8 80 80 DC 80 80 EC 0D
0F89: 80 80 92 80 80 FC 80 80 DB
0F91: F8 80 80 F8 81 80 F4 81 65
0F99: 80 CE 83 80 8E 87 80 9C 54

```

0FA1: 82 80 C8 81 80 86 80 80 F1
 0FA9: E0 80 80 D8 81 80 DC 81 3F
 0FB1: 80 EC 80 80 94 80 80 FC 08
 0FB9: 80 80 F8 80 80 B8 81 80 C9
 0FC1: DC 81 80 CE 83 80 8E 87 6E
 0FC9: 80 9C 82 80 C8 81 80 86 7B
 0FD1: 80 80 86 80 8F 80 9B 80 5F
 0FD9: 96 80 9C 80 9B 80 8F 80 7D
 0FE1: 8F 80 8F 80 8F 80 8F 80 FF
 0FE9: 87 80 87 80 82 80 8C 80 94
 0FF1: 98 80 BC 80 B6 80 9A 80 89
 0FF9: 8E 80 B6 80 BC 80 BC 80 40
 1001: BC 80 BC 80 BC 80 B8 80 19
 1009: B8 80 90 80 8C 80 F0 83 8B
 1011: 80 F8 87 80 F8 9F 80 F0 E0
 1019: 87 80 F8 87 80 F8 87 80 2C
 1021: F0 83 80 E0 83 80 F0 83 3C
 1029: 80 F8 87 80 FE 87 80 F8 D0
 1031: 83 80 F8 87 80 F8 87 80 42
 1039: F0 83 80 F0 81 80 FE 9F 7D
 1041: 80 FF BF 80 FF BF 80 FF A1
 1049: BF 80 FF BF 80 FE 9F 80 25
 1051: F0 83 80 F8 87 80 F8 87 22
 1059: 80 F8 87 80 F8 87 80 F8 D0
 1061: 87 80 F8 87 80 F8 87 80 74
 1069: B8 87 80 FC 80 80 FC 83 2B
 1071: 80 FC 87 80 F8 87 80 F8 E9
 1079: 87 80 F8 87 80 F8 83 80 84
 1081: FC 83 80 FC 87 80 FE 8F AC
 1089: 80 DF 8F 80 EE 87 80 FC 6F
 1091: 83 80 80 8F 80 F0 81 80 E7
 1099: F8 83 80 F8 87 80 F8 87 6E
 10A1: 80 F8 8F 80 F8 87 80 F8 1A
 10A9: 83 80 FC 83 80 FC 87 80 0B
 10B1: FE 8F 80 DF 8F 80 EE 87 27
 10B9: 80 FC 83 80 80 8F 80 F8 0E
 10C1: 80 80 FC 81 80 FE 81 80 7D
 10C9: FE 81 80 FF 81 80 FE 81 67
 10D1: 80 FC 81 80 FC 83 80 FE 9F
 10D9: 83 80 FF 87 80 DF 8F 80 77
 10E1: BE 87 80 FC 83 80 8F 80 E0
 10E9: 80 F0 81 80 FC 83 80 FE B4
 10F1: 83 80 FE 81 80 FE 81 80 6F
 10F9: FE 81 80 FC 81 80 FC 83 65
 1101: 80 FE 83 80 FF 87 80 DF 9A
 1109: 8F 80 BE 87 80 FC 83 80 E2
 1111: 8F 80 80 9E 80 BF 80 FF 19
 1119: 80 FE 80 FC 80 FF 80 BF DF
 1121: 80 BF 80 BF 80 BF 80 BF 43
 1129: 80 9F 80 9F 80 8E 80 BC 79


```

1131: 80 98 80 BC 80 B6 80 9A 10
1139: 80 8E 80 B6 80 BC 80 BC 6F
1141: 80 BC 80 BC 80 BC 80 B8 5F
1149: 80 B8 80 90 80 8C 80 94 BE
1151: 80 9C 80 A2 80 9C 80 9C 29
1159: 80 AA 80 9C 80 AA 80 9C 8C
1161: 80 AA 80 9C 80 AA 80 94 8C
1169: 80 BE 80 BE 80 FF 80 BE 3B
1171: 80 BE 80 FF 80 BE 80 FF 93
1179: 80 BE 80 FF 80 BE 80 FF 9B
1181: 80 BE 80 94 80 9C 80 A2 07
1189: 80 9C 80 88 80 88 80 9C 6F
1191: 80 AA 80 9C 80 AA 80 9C C4
1199: 80 AA 80 9C 80 AA 80 94 C4
11A1: 80 BE 80 BE 80 FF 80 BE 73
11A9: 80 9C 80 9C 80 BE 80 FF 0D
11B1: 80 BE 80 FF 80 BE 80 FF D3
11B9: 80 BE 80 FF 80 BE 80 7E 5A
11C1: 7F 00 7E 7F 00 06 60 00 44
11C9: 06 60 00 06 60 00 06 60 D6
11D1: 00 06 60 00 06 60 00 06 39
11D9: 60 00 06 60 00 06 60 00 CB
11E1: 06 60 00 06 60 00 06 60 EE
11E9: 00 06 60 00 06 60 00 06 51
11F1: 60 00 06 60 00 06 60 00 E3
11F9: 06 60 00 06 60 00 06 60 07
1201: 00 06 60 00 06 60 00 06 6A
1209: 60 00 06 60 00 06 60 00 FC
1211: 06 60 00 7E 7F 00 7E 7F B0
1219: 00 7F 7F 01 7F 7F 01 0F 28
1221: 70 01 0F 70 01 0F 70 01 CC
1229: 0F 70 01 0F 70 01 0F 70 18
1231: 01 0F 70 01 0F 70 01 0F 03
1239: 70 01 0F 70 01 0F 70 01 E4
1241: 0F 70 01 0F 70 01 0F 70 30
1249: 01 0F 70 01 0F 70 01 0F 1B
1251: 70 01 0F 70 01 0F 70 01 FC
1259: 0F 70 01 0F 70 01 0F 70 48
1261: 01 0F 70 01 0F 70 01 0F 33
1269: 70 01 0F 70 01 7F 7F 01 F4
1271: 7F 7F 01 CE 0E 20 20 20 93
1279: 20 20 20 2E 20 53 2E 20 67
1281: 20 03 08 27 10 2E 20 53 B6
1289: 52 20 7F 54 20 20 20 00 D5
1291: 20 FE 0E 20 53 20 20 3A DE
1299: 20 20 20 20 20 20 20 03 A0
12A1: 0F 3F 10 53 20 20 20 53 69
12A9: 20 20 50 2E 20 00 44 E6 43
12B1: 0E 00 20 20 45 2E 20 20 26
12B9: 20 20 20 20 20 03 08 0F 28

```

12C1: 10 20 53 20 4A 20 20 20 95
 12C9: 20 41 20 20 20 2B 0F 20 40
 12D1: 53 20 4A 20 41 53 20 56 E0
 12D9: 20 48 20 03 0E 6C 10 7F 16
 12E1: 20 20 20 20 20 20 36 20 32
 12E9: 20 2E 20 D3 0F 20 20 20 44
 12F1: 20 20 20 20 7F 20 20 4F 40
 12F9: 20 02 0F 14 11 20 20 20 3B
 1301: 00 20 20 20 50 33 20 33 F7
 1309: 20 55 0F 45 20 20 00 41 8D
 1311: 7F 20 20 49 20 44 20 03 EC
 1319: 0E 96 10 20 32 4E 20 20 1B
 1321: 20 20 20 20 20 20 20 7F A6
 1329: 0F 43 20 20 03 20 20 20 A6
 1331: 20 20 20 20 20 03 0E C0 5F
 1339: 10 20 33 20 20 20 20 41 DA
 1341: 20 33 20 33 20 F1 0F 20 82
 1349: 53 43 20 00 55 20 20 49 A2
 1351: 7F 20 00 02 0F 32 11 20 E2
 1359: 20 00 20 20 20 20 49 53 FC
 1361: 20 20 20 A9 0F 20 2E 20 B3
 1369: 52 2E 20 20 20 20 20 20 2C
 1371: 20 03 0E EA 10 20 7F 54 2D
 1379: 20 20 48 00 20 7F 20 52 52
 1381: 20 50 11 20 20 20 20 20 D1
 1389: 20 20 53 53 45 20 45 02 9E
 1391: 0D 6A 11 32 20 20 20 20 FF
 1399: 3B 20 20 20 20 53 2E 84 9A
 13A1: 11 20 20 20 00 53 20 20 0C
 13A9: 20 7F 20 7F 20 02 0F A2 85
 13B1: 11 20 53 43 52 20 20 53 AD
 13B9: 20 20 00 20 00 C0 11 20 3F
 13C1: 20 20 7F 20 7F 20 52 2E 41
 13C9: 20 33 20 03 1E 1A 12 20 9E
 13D1: 20 20 20 20 20 20 2E 20 14
 13D9: 52 2E 20 A9 00 8D FC 1B 9E
 13E1: A9 74 85 EE A9 12 85 EF 2A
 13E9: AD FC 1B C9 0C F0 2F A0 49
 13F1: 0E B1 EE 8D DD 1B C8 B1 E0
 13F9: EE 8D DE 1B 20 20 14 A5 D7
 1401: EE 18 69 10 85 EE 90 02 DF
 1409: E6 EF 20 20 14 18 A5 EE E1
 1411: 69 0E 85 EE 90 02 E6 EF 5B
 1419: EE FC 1B 4C E9 13 60 A9 26
 1421: 01 8D D5 1B AD D5 1B 0A 9E
 1429: A8 A5 EC 91 EE C8 A5 ED 99
 1431: 91 EE A0 00 B1 EE 85 FC 43
 1439: C8 B1 EE 85 FD AD DE 1B E7
 1441: 8D DF 1B A0 00 8C DB 1B 9A
 1449: A9 00 8D DC 1B B1 FC 8D EC

```

1451: D4 1B 0A AE D5 1B 0A 2E 34
1459: DC 1B CA D0 F9 2C D4 1B 62
1461: 10 02 38 24 18 6A 0D DB BB
1469: 1B 91 EC AD DC 1B 8D DB 46
1471: 1B C8 CC DD 1B D0 D1 18 AB
1479: A5 FC 6D DD 1B 85 FC 90 B8
1481: 02 E6 FD 1B A5 EC 6D DD 3F
1489: 1B 85 EC 90 02 E6 ED CE 9D
1491: DF 1B D0 AF AC D5 1B C8 41
1499: 8C D5 1B C0 07 F0 03 4C 3B
14A1: 25 14 60 A5 E6 C9 20 D0 37
14A9: 08 A9 03 A0 1C A2 00 F0 07
14B1: 06 A9 53 A0 1C A2 01 8E B7
14B9: FD 1B 85 1E 84 1F 60 A0 3C
14C1: 0E B1 EE 8D DD 1B C8 B1 B2
14C9: EE 8D DE 1B A0 00 AD D7 92
14D1: 1B 91 1E C8 AD D6 1B 91 CC
14D9: 1E C8 AD D5 1B 91 1E C8 7A
14E1: A5 EE 91 1E C8 A5 EF 91 FA
14E9: 1E C8 AD DE 1B 91 1E C8 1B
14F1: AD DD 1B 91 1E A5 1E 18 C0
14F9: 69 08 85 1E 90 02 E6 1F E4
1501: AE FD 1B FE FE 1B AD D5 EA
1509: 1B 0A 65 EE 85 FC A5 EF 3A
1511: 69 00 85 FD 1B A5 FC 69 3B
1519: 10 85 1A A5 FD 69 00 85 65
1521: 1B A0 00 B1 FC AA C8 B1 F1
1529: FC 86 FC 85 FD A0 00 B1 8F
1531: 1A AA C8 B1 1A 86 1A 85 EB
1539: 1B AD D7 1B 8D DA 1B AD C4
1541: DE 1B 8D DF 1B 20 EC 16 9A
1549: 20 B5 15 EE DA 1B A5 FC 0E
1551: 18 6D DD 1B 85 FC 90 02 93
1559: E6 FD A5 1A 18 6D DD 1B 1A
1561: 85 1A 90 02 E6 1B CE DF 28
1569: 1B D0 DA 60 AE FD 1B BD 18
1571: FE 1B F0 3F A0 00 B1 1E 7A
1579: 8D D7 1B C8 B1 1E 8D D6 48
1581: 1B C8 B1 1E 8D D5 1B C8 46
1589: B1 1E 85 EE C8 B1 1E 85 82
1591: EF C8 B1 1E 8D DE 1B C8 E4
1599: B1 1E 8D DD 1B 20 C7 15 B1
15A1: A5 1E 18 69 08 85 1E 90 E2
15A9: 02 E6 1F AE FD 1B DE FE 76
15B1: 1B D0 C1 60 AC DD 1B 88 77
15B9: B1 FE 31 1A 51 FE 11 FC E9
15C1: 91 FE 88 10 F3 60 AD D7 DA
15C9: 1B 8D DA 1B AD DE 1B 8D 9E
15D1: DF 1B 20 EC 16 AC DD 1B BF
15D9: 88 B1 1C 91 FE 88 10 F9 85

```



```

15E1: EE DA 1B CE DF 1B D0 EA 82
15E9: 60 A9 00 8D F5 1B A9 04 FA
15F1: 8D D8 1B AD D8 1B 8D D7 7D
15F9: 1B A9 00 8D D9 1B A9 BE 42
1601: 38 ED D8 1B 8D DE 1B 20 CF
1609: B2 16 A9 26 8D D7 1B 38 E5
1611: A9 BE ED D8 1B 8D D9 1B EA
1619: AD D8 1B 8D DE 1B 20 B2 E4
1621: 16 AD F5 1B F0 23 A9 00 9B
1629: 8D DA 1B 8D D6 1B A9 C0 46
1631: 8D DF 1B 20 EC 16 A0 1D 9F
1639: B1 FE 91 1C 88 10 F9 EE 59
1641: DA 1B CE DF 1B D0 EC F0 60
1649: 12 38 AD E5 1B E9 04 8D B6
1651: E5 1B 38 AD EE 1B E9 04 D4
1659: 8D EE 1B 20 A4 14 AE FD 3E
1661: 1B A9 00 9D FE 1B 20 3C 40
1669: 0D 20 45 0C 20 81 0D 20 CE
1671: 61 1A 20 5A 17 AD D8 1B BA
1679: C9 26 B0 22 69 04 8D D8 9B
1681: 1B 38 E9 26 90 15 48 18 6A
1689: 6D E5 1B 8D E5 1B 68 18 A6
1691: 6D EE 1B 8D EE 1B A9 26 C9
1699: 8D D8 1B 4C F4 15 AD F5 38
16A1: 1B D0 08 A9 01 8D F5 1B 70
16A9: 4C F4 15 A9 FF 8D E9 1B 9B
16B1: 60 AD DE 1B 8D DF 1B A9 D2
16B9: 00 8D D6 1B AD D7 1B 8D 66
16C1: DA 1B 20 EC 16 A5 1C 85 F9
16C9: FC A5 1D 85 FD AD D9 1B 4F
16D1: 8D DA 1B 20 EC 16 A0 1D FE
16D9: B1 FC 91 FE 88 10 F9 EE A7
16E1: D7 1B EE D9 1B CE DF 1B 2B
16E9: D0 D2 60 AD DA 1B 29 3F EE
16F1: A8 B9 1A 17 05 E6 85 FF 64
16F9: 09 60 85 1D AD DA 1B 29 7D
1701: 08 F0 02 A9 80 18 2C DA E1
1709: 1B 70 04 10 04 69 28 69 E1
1711: 28 6D D6 1B 85 FE 85 1C 8A
1719: 60 00 04 08 0C 10 14 18 59
1721: 1C 00 04 08 0C 10 14 18 3F
1729: 1C 01 05 09 0D 11 15 19 C6
1731: 1D 01 05 09 0D 11 15 19 4F
1739: 1D 02 06 0A 0E 12 16 1A D6
1741: 1E 02 06 0A 0E 12 16 1A 5F
1749: 1E 03 07 0B 0F 13 17 1B E6
1751: 1F 03 07 0B 0F 13 17 1B 6F
1759: 1F A5 E6 C9 40 A9 00 2A CC
1761: AA BD 54 C0 A5 E6 8D E0 AF
1769: 1B 49 60 85 E6 60 A9 60 48

```

1771: 85 E6 20 F2 F3 A9 00 8D 23
1779: DA 1B 8D D6 1B 20 EC 16 44
1781: A9 00 8D FC 1B 20 B0 17 D7
1789: C9 05 D0 F9 EE DA 1B EE 9F
1791: DA 1B 20 EC 16 20 B0 17 70
1799: C9 0A D0 F9 EE DA 1B EE F0
17A1: DA 1B AD DA 1B C9 BA B0 8D
17A9: 03 4C 7E 17 4C D9 17 AC 52
17B1: FC 1B B9 C5 17 48 B9 CF D5
17B9: 17 A8 68 91 FE EE FC 1B 8C
17C1: AD FC 1B 60 08 01 40 20 54
17C9: 01 20 04 02 01 04 01 08 43
17D1: 0E 15 1C 01 08 0F 16 1C A4
17D9: A9 00 8D D6 1B 8D FC 1B 20
17E1: AC FC 1B C0 05 F0 1B B9 EA
17E9: 59 18 8D DA 1B 20 03 18 A1
17F1: 20 15 18 20 15 18 20 03 C6
17F9: 18 EE FC 1B 4C E1 17 4C A5
1801: 5E 18 20 EC 16 A0 00 B9 26
1809: 4F 18 91 FE C8 C0 02 D0 27
1811: F6 4C 24 18 20 EC 16 A0 57
1819: 00 B9 51 18 91 FE C8 C0 3E
1821: 08 D0 F6 A5 FE A6 FF 85 DA
1829: FC 86 FD 98 18 65 FE 85 9C
1831: FE A0 00 B1 FC 91 FE C8 19
1839: 98 18 65 FE 29 7F C9 1E 51
1841: F0 08 C9 46 F0 04 C9 6E 23
1849: D0 E9 EE DA 1B 60 D5 AA 98
1851: D0 A0 C1 82 85 8A 94 A8 9A
1859: 22 48 6E 94 BA A9 19 8D FF
1861: D6 1B A9 00 8D FC 1B 8D 1D
1869: DA 1B AC FC 1B C0 05 F0 0A
1871: 1B 20 92 18 38 A9 1C ED 99
1879: D6 1B 8D D6 1B 18 AD DA 6A
1881: 1B 69 06 8D DA 1B EE FC 51
1889: 1B 4C 6B 18 60 84 82 D0 34
1891: 80 A2 00 A9 8E 85 FC A9 73
1899: 18 85 FD A9 01 8D DE 1B A8
18A1: 20 EC 16 A0 01 B1 FC 91 44
18A9: FE 88 10 F9 18 A5 FC 69 D7
18B1: 02 85 FC 90 02 E6 FD EE 83
18B9: DA 1B CE DE 1B 10 E1 E8 AB
18C1: E0 10 D0 CF 60 AD E1 1B 16
18C9: 0A 0A 38 6D E1 1B 8D E1 D7
18D1: 1B 60 80 BC 98 BC BC B0 65
18D9: FE BC FE BC BC FC BE BC 78
18E1: BE FE FE BC E6 98 E0 E6 1F
18E9: 86 E6 BE BC BE BC BE BC DD
18F1: FE E6 E6 E6 E6 E6 FE 80 F7
18F9: E6 9C E6 E6 B8 86 86 E0 DD

1901: E6 E6 E6 E6 E6 E6 86 86 12
 1909: E6 E6 98 E0 E6 86 FE E6 BF
 1911: E6 E6 E6 E6 E6 98 E6 E6 0A
 1919: E6 E6 E6 B0 80 F6 98 B0 22
 1921: B0 B4 BE BE B0 BC E6 E6 C9
 1929: E6 86 E6 86 86 86 E6 98 6A
 1931: E0 B6 86 E6 E6 E6 E6 E6 48
 1939: E6 8C 98 E6 E6 E6 E6 E6 0B
 1941: 98 80 EE 98 8C E0 FE E0 0E
 1949: E6 98 E6 FC FE BE 86 E6 A8
 1951: BE BE F6 FE 98 E0 9E 86 6D
 1959: E6 E6 E6 BE E6 BE B0 98 AD
 1961: E6 E6 E6 BC BC 8C 80 E6 69
 1969: 98 E6 E6 B0 E6 E6 8C E6 5C
 1971: B0 E6 E6 E6 E6 86 86 E6 46
 1979: E6 98 E6 E6 86 E6 E6 E6 15
 1981: 86 B6 E6 E6 98 E6 E6 FE 1D
 1989: E6 98 86 80 BC BC FE BC C1
 1991: B0 BC BC 8C BC 98 E6 FE C0
 1999: BE BE FE 86 BE E6 98 BC A2
 19A1: E6 FE E6 E6 BC 86 EC E6 13
 19A9: BE 98 BE 98 E6 E6 98 FE C5
 19B1: AD F8 1B 85 EE AD F9 1B F1
 19B9: 85 EF A5 EE 05 EF D0 01 D8
 19C1: 60 A0 04 A9 01 AE FD 1B 41
 19C9: F0 01 0A 31 EE F0 4B B1 8C
 19D1: EE 30 05 2C 34 1B F0 4D 28
 19D9: C8 B1 EE 8D DA 1B C8 B1 1A
 19E1: EE 8D D6 1B A5 EE 18 69 FD
 19E9: 07 85 FC A5 EF 69 00 85 A5
 19F1: FD A9 06 8D DF 1B 20 EC BF
 19F9: 16 A0 00 B1 1C 91 FE C8 68
 1A01: B1 FC D0 F7 EE DA 1B CE CE
 1A09: DF 1B D0 EA A9 01 AE FD 69
 1A11: 1B F0 01 0A A0 04 51 EE 76
 1A19: 91 EE B1 EE 2C 34 1B F0 50
 1A21: 04 29 03 F0 06 20 EF 1A BB
 1A29: 4C BB 19 A0 02 B1 EE 85 D9
 1A31: FC C8 B1 EE 85 FD 20 EF 8F
 1A39: 1A A0 00 A5 EE 91 FC C8 7D
 1A41: A5 EF 91 FC C8 05 EE D0 4F
 1A49: 0B A5 FC 8D FA 1B A5 FD 72
 1A51: 8D FB 1B 60 A5 FC 91 EE E7
 1A59: C8 A5 FD 91 EE 4C BB 19 6D
 1A61: AD F8 1B 85 EE AD F9 1B A3
 1A69: 85 EF A5 EE 05 EF D0 01 8A
 1A71: 60 A0 04 B1 EE 2C 34 1B 45
 1A79: D0 6E C9 00 30 0C A9 01 F0
 1A81: AE FD 1B F0 01 0A 31 EE 80
 1A89: D0 5E C8 B1 EE 8D DA 1B 70


```

1A91: C8 B1 EE 8D D6 1B A9 D2 96
1A99: 8D BE 1A A9 18 8D BF 1A B2
1AA1: A9 06 8D DF 1B A5 EE 18 41
1AA9: 69 07 85 FC A5 EF 69 00 94
1AB1: 85 FD 20 EC 16 A0 00 B1 DF
1AB9: FC F0 09 AA BD FF FF 91 F3
1AC1: FE C8 D0 F3 EE DA 1B AD C7
1AC9: BE 1A 18 69 25 8D BE 1A 74
1AD1: 90 03 EE BF 1A CE DF 1B CF
1AD9: D0 D8 A9 01 AE FD 1B F0 86
1AE1: 01 0A A0 04 11 EE 91 EE C3
1AE9: 20 EF 1A 4C 6B 1A A0 00 37
1AF1: B1 EE AA C8 B1 EE 86 EE E1
1AF9: 85 EF 60 48 98 48 AD FA B9
1B01: 1B 85 EE AD FB 1B 85 EF 26
1B09: A0 00 68 91 EE C8 68 91 B2
1B11: EE 20 EF 1A A9 00 A8 91 96
1B19: EE C8 91 EE C8 AD FA 1B 28
1B21: 91 EE C8 AD FB 1B 91 EE 2E
1B29: A5 EE 8D FA 1B A5 EF 8D 2C
1B31: FB 1B 60 40 A9 E8 8D 87 CF
1B39: 1B A9 F4 8D 7D 1B A9 1E A8
1B41: 8D 6A 1B 4C 56 1B A9 E0 54
1B49: 8D 87 1B A9 E8 8D 7D 1B B9
1B51: A9 1F 8D 6A 1B A9 01 8D 8B
1B59: 00 1C A0 00 AD 87 1B 8D F9
1B61: 02 1C 4E 00 1C 90 0C B9 5E
1B69: 00 1E C8 8D 01 1C A9 80 65
1B71: 8D 00 1C 4E 01 1C 90 03 73
1B79: AD 30 C0 A2 FF E8 D0 FD 18
1B81: 90 03 AD 30 C0 A2 FF E8 F2
1B89: D0 FD EE 02 1C D0 D3 18 89
1B91: AD 87 1B E9 01 8D 87 1B EA
1B99: AD 7D 1B 69 01 8D 7D 1B 54
1BA1: 90 BA 60 A9 88 8D 00 1E 0E
1BA9: A0 01 B9 FF 1D 99 00 1E 15
1BB1: C8 D0 F7 A9 80 A0 03 99 40
1BB9: 00 1F 88 10 FA A9 AA A0 3E
1BC1: 03 99 04 1F 88 10 FA A0 6D
1BC9: 08 B9 F8 1E 99 00 1F C8 47
1BD1: D0 F7 60 80 D5 8A 80 D5 32

```

The Witching Hour

Brian Flynn

Apple version by Kevin Martin

Witches and ghosts make this game of skill and foresight ideal for a dark and stormy night. Play it on any Apple II series computer using DOS 3.3 or ProDOS—just make sure the lights are on.

When autumn winds send a shiver down your spine and the witching hour draws near, there's no better entertainment than a good computer game. "The Witching Hour" is an absorbing contest of strategy based on Alquerque, a board game played in ancient Egypt and still popular in Spain today. Type in and save The Witching Hour, then get ready for some ghostly and ghastly action.

Broomsticks and Sheets

The Witching Hour pits broomstick-straddling witches against ethereal ghosts and is played on a board of 25 squares with 12 pieces to a side. After choosing sides, you attempt to take your opponent's players by jumping over them. You can move vertically, horizontally, or diagonally. However, certain diagonal moves are illegal (the lines between squares show where you can go), and only one square is vacant when the game begins.

Jumping an opposing player's piece removes that piece from the board. If no capture is possible, you may move any piece to an adjacent empty square. *You may not pass up a capture*—if it's possible to jump an opponent, you must always do so. Don't worry. The computer won't let you make illegal moves even if you're playing against a friend.

If you jump one opposing piece and there's another jump possible, you'll be asked if you want to take the move. Press Y for yes or N for no. Answering yes means you'll have to provide another destination square. When you're playing a game against the computer, it will *always* take this second (or even third) jump—which could be useful in certain situations.

Play ends when all the pieces from one side have been removed from the board. You can play against a friend or measure your skills against the computer. Like other contests of strategy, The Witching Hour is simple to learn, yet a challenge to master. Hint: It's sometimes smart to sacrifice a player to draw the opponent into a dangerous position.

When the program starts, you must choose between a one- or two-player game. After the game board is drawn, a flashing box shows which square you're on. Move this box with keyboard controls. Press the I key to go up, J to go left, K for down, and L for right. Press the Return key when the box is on the piece you want to move, then move the box to the desired square and press Return again.

The Witching Hour

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
90 10 Z = 1: HOME : TEXT
6E 11 POKE 232,96: POKE 233,3: SCALE= 27: ROT= 0: HC
    OLOR= 3
EA 20 VTAB 8: HTAB 11: PRINT "THE WITCHING HOUR"
17 30 VTAB 12: HTAB 6: PRINT "PRESS '1' FOR ONE PLAY
    ER"
92 40 HTAB 12: PRINT "'2' FOR TWO PLAYERS"
D1 50 GET A$:NP = ASC (A$) - 48: IF NP < 1 OR NP > 2
    THEN 50
B6 60 IF NP = 2 THEN 110
EB 70 HTAB 5: PRINT "DO YOU WANT TO GO FIRST (Y/N)?"
    ;
2D 80 GET A$:F1 = 1: IF A$ < > "Y" AND A$ < > "N" TH
    EN 80
38 90 IF A$ = "N" THEN F1 = - 1
96 110 HOME : HGR : GOSUB 650: GOTO 350
D2 120 H = 0:K = 0: FOR A = 7 TO 35: GOSUB 160: NEXT
BE 130 GOSUB 270: IF H < 1 THEN 340
E5 140 H = 0:K = 0:A = T: GOSUB 160: IF H < 1 THEN 3
    40
97 150 GOTO 130
16 160 IF B(A) = 0 OR B(A) = - S OR B(A) = 2 THEN RE
    TURN
E5 170 FOR B = 0 TO D(A - 7):C = A + M(B): IF B(C) =
    S OR B(C) = 2 THEN 260
F1 180 IF B(C) THEN 220
1C 190 SC = RND (0) * .9: IF H < SC THEN H = SC:F =
    A:T = C
A6 200 IF CK = 1 AND T1 = C THEN L = 1:B = 7
14 210 GOTO 260
52 220 IF B(C + M(B)) THEN 260
BF 230 SC = 1 + RND (0) * .9: IF H < SC THEN H = SC:
    F = A:T = C + M(B):K = C
D8 240 IF CK = 0 THEN 260
EF 250 IF T1 = C + M(B) THEN L = 1:K1 = C:B = 7
CA 260 NEXT : RETURN
1A 270 A = F:B(T) = B(F):B(F) = 0: GOSUB 1130
```



```

39 280 IF K THEN B(K) = 0:A = K: GOSUB 1130
04 290 A = T: GOTO 1130
5C 300 GOSUB 610: IF S = 1 THEN VTAB 21: HTAB 12: PR
INT "THE WITCHES WIN!": GOTO 320
0A 310 VTAB 21: HTAB 12: PRINT "THE GHOSTS WIN!"
5E 320 HTAB 10: PRINT "PRESS THE <SPACEBAR>"
76 330 GET A$: IF A$ < > " " THEN 330
AF 331 RUN
D7 340 S = - S:Z = - (Z = 0):H = 0: FOR A = 7 TO 35:
GOSUB 160: NEXT : IF H = 0 THEN 300
44 350 D = 0: GOSUB 610: IF NP = 1 AND S = - 1 THEN
Z = 1
EA 360 IF F1 = - 1 THEN Z = 1
35 370 IF NP = 1 AND S = F1 THEN 120
4F 380 IF S = 1 THEN VTAB 21: HTAB 12: PRINT "THE GH
OST'S TURN": GOTO 400
66 390 VTAB 21: HTAB 12: PRINT "THE WITCH'S TURN"
E1 400 VTAB 22: HTAB 12: PRINT "FROM: "; CHR$ (8);
DA 410 GOSUB 1250
B7 420 PRINT A$:A = N( ASC (A$) - 65):Z = A
DC 430 HTAB 25: PRINT "TO: "; CHR$ (8);
E0 440 GOSUB 1250
E9 450 PRINT A$:T1 = N( ASC (A$) - 65):CK = 1:L = 0
:K1 = 0: GOSUB 160:CK = 0
DB 451 H = 0:A = 7
62 452 IF A = 36 THEN 460
64 453 GOSUB 160: IF H > = 1 THEN 460
B1 454 A = A + 1: IF A < 36 THEN 453
BC 460 IF D THEN 540
10 470 IF L THEN 545
96 530 GOSUB 620: GOTO 350
7D 540 IF L = 0 OR K1 = 0 THEN GOSUB 620: GOTO 570
A9 545 IF K1 = 0 AND H > = 1 THEN 530
45 550 F = Z:T = T1:K = K1: GOSUB 270: IF K1 = 0 THE
N 340
97 560 A = T:Z = A:H = 0: GOSUB 160: IF H < 1 THEN 3
40
18 570 GOSUB 610: VTAB 21: HTAB 11: PRINT "JUMP AGAI
N (Y/N)?";
62 580 GET A$: IF A$ < > "Y" AND A$ < > "N" THEN 580
31 590 GOSUB 610: IF A$ = "N" THEN S = - S: GOTO 350
72 600 D = 1: VTAB 22: GOTO 430
4F 610 PRINT : VTAB 21: FOR J = 1 TO 2: FOR I = 1 TO
40: PRINT " "; NEXT I,J
1C 611 RETURN
03 620 PRINT CHR$ (7);: RETURN
14 630 HOME : RETURN
E7 650 DIM D(28),B(42),X(35),Y(35),N(28)
DE 660 S = - 1: FOR A = 0 TO 7: READ M(A): NEXT : FO
R A = 0 TO 28: READ D(A): NEXT

```

```

53 670 FOR A = 0 TO 4: FOR F = 0 TO 4: H = 6 * A + F
    + 7: X(H) = 4 * F + 11: Y(H) = 4 * A: N(G) = H: G
    = G + 1: NEXT F, A
58 680 FOR A = 0 TO 42: READ B(A): NEXT : FOR A = 0
    TO 6: READ F: POKE 864 + A, F: NEXT : GOSUB 76
    0: GOSUB 1190: FOR A = 0 TO 42: GOSUB 1130: N
    EXT : RETURN
C8 690 DATA -6,1,6,-1,-5,7,5,-7
44 700 DATA 7,3,7,3,7,0,3,7,3,7,3,0
1A 710 DATA 7,3,7,3,7,0,3,7,3,7,3,0,7,3,7,3,7
A9 720 DATA 2,2,2,2,2,2,2,-1,-1,-1,-1,-1,2
44 730 DATA -1,-1,-1,-1,-1,2,-1,-1,0,1,1,2
BE 740 DATA 1,1,1,1,1,2,1,1,1,1,1,2,2,2,2,2,2,2
38 750 DATA 1,0,4,0,44,62,0
A2 760 FOR A = 768 TO 855: READ F: POKE A, F: NEXT
7F 770 POKE 6,0: POKE 7,141: IF PEEK (191 * 256) = 7
    6 THEN PRINT CHR$ (4); "PR#A$300": GOTO 790
C8 780 POKE 54,0: POKE 55,3: CALL 1002
E3 790 FOR A = 36352 TO 36567: READ F: POKE A, F: NEX
    T
18 800 RETURN
2C 1130 IF B(A) = 2 THEN RETURN
22 1140 VTABLE Y(A) + 1: HTAB X(A)
45 1150 IF B(A) < 0 THEN PRINT "QAB": HTAB X(A): PRI
    NT "FGH": HTAB X(A): PRINT "LMN"
38 1160 IF B(A) > 0 THEN PRINT "CDE": HTAB X(A): PRI
    NT "IJK": HTAB X(A): PRINT "OPQ"
AB 1170 IF B(A) = 0 THEN PRINT "RST": HTAB X(A): PRI
    NT "UVW": HTAB X(A): PRINT "XYZ"
F3 1180 RETURN
EA 1190 HCOLOR= 3
71 1200 FOR A = 11 TO 139 STEP 32: HPLOT 78,A TO 190
    ,A: NEXT
6A 1210 FOR A = 78 TO 190 STEP 28: HPLOT A,11 TO A,1
    39: NEXT
F8 1220 HPLOT 78,11 TO 194,140: HPLOT 194,11 TO 78,1
    40
39 1230 HPLOT 78,76 TO 136,11 TO 194,76 TO 136,140 T
    O 78,76
2E 1249 RETURN
D2 1250 F = 2: T1 = 2: QS = 2
2B 1260 SCALE= QS
83 1270 XDRAW 1 AT (T1 * 4 + 10) * 7 - 3, (F * 4) * 8
    + 25
3E 1275 PRINT CHR$ (F * 5 + T1 + 65); CHR$ (8);
49 1280 A$ = "": IF PEEK ( - 16384) > 128 THEN GET A
    $
AF 1285 XDRAW 1 AT (T1 * 4 + 10) * 7 - 3, (F * 4) * 8
    + 25
F8 1287 QS = QS + 5: IF QS > 27 THEN QS = 2

```

```

EE 1290 IF A$ = "I" AND F > 0 THEN F = F - 1
01 1291 IF A$ = "K" AND F < 4 THEN F = F + 1
BB 1292 IF A$ = "J" AND T1 > 0 THEN T1 = T1 - 1
B7 1293 IF A$ = "L" AND T1 < 4 THEN T1 = T1 + 1
A2 1300 IF A$ < > CHR$ (13) THEN 1260
40 1400 A$ = CHR$ (F * 5 + T1 + 65): RETURN
D6 1500 DATA 216,120,133,69,134,70
2E 1510 DATA 132,71,166,7,10,10
44 1520 DATA 176,4,16,62,48,4
BB 1530 DATA 16,1,232,232,10,134
60 1540 DATA 27,24,101,6,133,26
A3 1550 DATA 144,2,230,27,165,40
95 1560 DATA 133,8,165,41,41,3
01 1570 DATA 5,230,133,9,162,8
3E 1580 DATA 160,0,177,26,36,50
89 1590 DATA 48,2,73,127,164,36
47 1600 DATA 145,8,230,26,208,2
9F 1610 DATA 230,27,165,9,24,105
0F 1620 DATA 4,133,9,202,208,226
87 1630 DATA 165,69,166,70,164,71
72 1640 DATA 88,76,240,253
71 1700 DATA 255,129,129,129,129,129
F0 1710 DATA 139,171,255,128,128,192
CB 1720 DATA 192,208,212,224,255,192
6F 1730 DATA 192,192,194,202,234,199
9F 1740 DATA 255,129,129,225,129,225
60 1750 DATA 225,225,255,128,128,135
8C 1760 DATA 159,255,238,255,255,192
C9 1770 DATA 192,192,192,192,192,192
8D 1780 DATA 171,171,171,169,129,129
A0 1790 DATA 193,199,229,181,181,165
90 1800 DATA 168,170,170,170,199,193
48 1810 DATA 193,193,193,195,199,204
D1 1820 DATA 129,129,159,255,255,199
B9 1830 DATA 193,193,159,142,142,255
C1 1840 DATA 255,191,191,255,192,248
0C 1850 DATA 255,255,241,192,192,192
55 1860 DATA 223,223,255,159,135,129
72 1870 DATA 129,255,170,170,170,168
0F 1880 DATA 170,139,128,255,216,240
69 1890 DATA 255,193,192,192,192,255
7E 1900 DATA 129,129,129,129,129,129
13 1910 DATA 129,255,255,254,252,248
17 1920 DATA 248,224,128,255,193,193
47 1930 DATA 193,192,193,207,254,255
39 1940 DATA 127,1,1,1,1,1
46 1950 DATA 1,1,127,0,0,0
D2 1960 DATA 0,0,0,0,127,64
25 1970 DATA 64,64,64,64,64,64
A4 1980 DATA 1,1,1,1,1,1

```



```

53 1990 DATA 1,1,0,0,0,0
20 2000 DATA 0,0,0,0,64,64
FB 2010 DATA 64,64,64,64,64,64
7B 2020 DATA 1,1,1,1,1,1
3B 2030 DATA 1,127,0,0,0,0
FF 2040 DATA 0,0,0,127,64,64
F6 2050 DATA 64,64,64,64,64,127

```

Miami Ice

Jeff Kulczycki

Apple version by Tim Victor

Forget the flamingos, the sun, the powerboats, and the designer clothes. "Miami Ice" doesn't have a smuggler within miles. What it does have is ice—lots and lots of slick ice under your wheels. This action game challenges both your driving skills and powers of concentration. For the Apple II family of personal computers, using DOS 3.3 or ProDOS.

Ah, Miami—sun city of the South. A sparkling metropolis blessed with a tropical climate, palm trees, beaches, revived art deco architecture, stylish pastels, and classy elegance. Almost paradise.

You wake up on another bright, sunny Miami morning, sip a glass of freshly squeezed orange juice, don your white linen suit and sunglasses, and stroll outside—then get the shock of your life.

What's going on here? Overnight, a freak shift in the jet stream has piped a blistering cold front down from Ohio. The weather reporter had predicted a brief shower last evening, but that's not what happened. Instead, the Florida peninsula was blasted by the worst ice storm in 400 years. The Everglades are frozen solid. Pink flamingos are blue. And the streets of Miami are coated with a shimmering layer of slippery ice.

As you start your car—the pampered engine coughs and sputters in the bitter cold—you wonder what it's going to be like driving to work. A Miami native, you've never driven on ice before. In fact, you've never even *seen* this much ice since your boss's retirement party last year, when the caterers made that life-size ice sculpture of Ponce de Leon. You've heard the horror stories told by tourists about winter driving conditions up North, but never thought it could happen to you—not here, in Miami.

The minute you pull out onto the street, your worst fears come true. When you step on the gas pedal, the wheels spin and the car accelerates sluggishly. When you turn the steering wheel, the car slides all over the road. And when you step on the brakes—well, forget it.

You realize, desperately, that you've got to make it to the parking garage across town without smashing your car to smithereens. It won't be easy. But at least there's one thing in your favor—you've got the whole road to yourself. Everyone else, it seems, had the good sense to stay home.

Out of Control

Control your car with a joystick or paddles—the choice is yours. The object of this topsy-turvy Miami game is to drive your car over ice-covered streets to reach the safety of a garage. The joystick or paddle button is the gas pedal; pushing the stick right or left (or twisting the paddle knob) steers the car.

Here's the twist—the car doesn't respond instantly to your commands. It tends to slide in the same direction even after you've steered it in another direction. Then, when you try to recover, you often overcorrect and start sliding in yet another new direction. It's an inertial nightmare—much like real winter driving.

When you hit a guardrail or some other obstruction, your car cracks up. You get three cars per game. If you reach the safety of the garage, the game isn't over. Instead, you advance to another screen whose streets are even harder to navigate.

The number of points you score depends on how soon you reach the garage. As an incentive to recklessness, an invisible timer starts counting down when you begin each new screen. If you reach the garage, you score the number of points left on the timer. If the timer runs out, you can still reach the garage, but you won't get any points. You will, however, advance to the next screen. (The only way you'll know if the timer's run out is if you *do* reach the next level and you're not awarded any points.)

Machine Language Slick

"Miami Ice" is written completely in machine language and must be entered with the "Apple MLX" machine language entry program found in Appendix C. Be sure you read and understand the instructions for using MLX before you begin entering the data below.

When you first run MLX, you'll be asked for a starting and an ending address. Here are the addresses you'll need for Miami Ice:

Starting address: 1000

Ending address: 1E97

After you've typed in all the data, be sure to save at least one copy before you exit MLX. To start Miami Ice, type BRUN "filename" (where *filename* is the name you used when you saved the Miami Ice data), then press Return.

Plug in a joystick or paddles. To reach the garage safely and advance to the next screen, you have to enter the front of the garage without bumping into any of its walls. There are seven screens in all. The game normally starts at screen 1, but you can begin a new game at any screen by pressing the controller button to change the screen number. This lets you skip the easier screens as you become a better ice driver or peek at the hardest screens while you're still a beginner.

Miami Ice

For mistake-proof program entry, use "Apple MLX" (Appendix C) to type in this program.

START ADDRESS: 1000

END ADDRESS: 1E97

```

1000: A9 00 85 EC A9 60 85 ED 3C
1008: A9 1A 85 FA A9 1A 85 FB A0
1010: 20 B3 17 20 78 17 A9 FF 85
1018: 8D C4 1E 20 97 18 A9 03 69
1020: 8D BB 1E A9 00 8D BF 1E 28
1028: 8D C0 1E F0 03 20 7E 19 C0
1030: 2C 57 C0 2C 52 C0 2C 54 59
1038: C0 2C 50 C0 20 B2 14 A9 77
1040: 00 A0 0A 91 FA A0 11 91 F0
1048: FA 2C 54 C0 A9 20 85 E6 47
1050: A9 40 8D 97 1E A9 01 8D A7
1058: 98 1E A9 01 8D 1D 1A A9 50
1060: 01 8D 1A 1A A9 01 8D 1B D0
1068: 1A A9 0A 8D 1C 1A A9 02 B8
1070: 8D 1E 1A A9 00 8D B1 1E 74
1078: A9 00 8D AF 1E 8D B0 1E C0
1080: 20 DA 12 AD 61 C0 30 FB EE
1088: 20 90 18 EE B9 1E D0 26 DC
1090: C9 78 90 15 C9 89 90 1E CA
1098: EE 1E 1A AD 1E 1A C9 08 CA
10A0: D0 11 A9 00 8D 1E 1A F0 AC
10A8: 0A CE 1E 1A 10 05 A9 07 D5
10B0: 8D 1E 1A 20 DA 12 AD 61 40
10B8: C0 10 CD 8D A5 1E AE 1E F0
10C0: 1A BD E3 12 8D A6 1E BD FB
10C8: EB 12 8D A7 1E A9 01 8D B6

```

10D0: A8 1E A9 00 8D B2 1E A9 1F
10D8: 00 8D C1 1E 8D C2 1E 20 4A
10E0: 90 18 C9 78 90 08 E9 10 98
10E8: C9 79 B0 02 A9 78 18 69 4B
10F0: 08 4A 4A 4A 4A 38 E9 08 A4
10F8: 8D A9 1E 48 AD A8 1E 69 4B
1100: F8 2C A9 1E 30 0C 49 FF 05
1108: 38 69 00 CD A9 1E B0 0A AE
1110: 90 05 CD A9 1E 90 03 8D D6
1118: A9 1E 68 F0 19 38 ED A9 E1
1120: 1E F0 0A 30 04 A9 02 D0 6D
1128: 0D A9 FE D0 09 A9 01 2C 45
1130: A9 1E 10 02 A9 FF 8D AA E3
1138: 1E AD 61 C0 10 0F 2C A5 C7
1140: 1E 30 0A A0 08 CC A8 1E AB
1148: 90 03 EE A8 1E 8D A5 1E 6C
1150: AD A9 1E AC A8 1E 20 FD 3E
1158: 12 A5 51 48 AC A6 1E 20 F7
1160: F3 12 AD A7 1E 18 65 50 9D
1168: 8D A7 1E 68 AC A7 1E 20 E5
1170: F3 12 AD A6 1E 38 E5 50 1F
1178: 8D A6 1E AD A8 1E AC A6 67
1180: 1E 20 FD 12 A5 50 8D AB CF
1188: 1E A5 51 8D AC 1E AD A8 08
1190: 1E AC A7 1E 20 FD 12 A5 86
1198: 50 8D AD 1E A5 51 8D AE 1A
11A0: 1E AC B2 1E C8 D0 0C A0 77
11A8: 00 AD A8 1E C9 01 F0 03 64
11B0: CE A8 1E 8C B2 1E EE C1 9E
11B8: 1E D0 0A EE C2 1E D0 05 83
11C0: A9 FF 8D C2 1E 18 AD AF F1
11C8: 1E 6D AC 1E 8D AF 1E AD E1
11D0: 1B 1A 6D AB 1E C9 07 2C C1
11D8: AB 1E 30 09 90 0E EE 1A A3
11E0: 1A E9 07 B0 07 90 05 CE C9
11E8: 1A 1A 69 06 18 6D B1 1E 24
11F0: C9 07 90 05 E9 07 EE 1A 7F
11F8: 1A 8D 1B 1A 4D 1A 1A 29 C0
1200: 01 8D B1 1E F0 11 CE 1B A4
1208: 1A 10 0C AD 1B 1A 18 69 74
1210: 07 8D 1B 1A CE 1A 1A 18 4B
1218: AD B0 1E 6D AE 1E 8D B0 93
1220: 1E AD 1C 1A 6D AD 1E 8D CF
1228: 1C 1A AD A6 1E C9 20 90 E9
1230: 0A C9 E0 B0 06 C9 00 10 5A
1238: 0D 30 21 A0 00 2C A7 1E 3B
1240: 30 2E A0 04 D0 2A AD A7 8E
1248: 1E A0 02 C9 E0 B0 21 C9 56
1250: 20 90 1D C9 00 10 02 88 B5
1258: 24 C8 D0 14 AD A7 1E A0 05

```

1260: 06 C9 20 90 0B C9 E0 B0 F8
1268: 07 C9 00 10 02 C8 24 88 97
1270: 98 18 6D AA 1E 10 03 18 8E
1278: 69 0B C9 08 90 02 E9 08 75
1280: 8D 1E 1A 20 DA 12 AD 88 6B
1288: 1E F0 37 AD BA 1E F0 35 1F
1290: A9 20 38 ED C2 1E 90 1E 45
1298: 0A 0A 8D C5 1E AE B3 1E 83
12A0: AD C5 1E 18 6D BF 1E C9 C2
12A8: 64 90 05 EE C0 1E E9 64 69
12B0: 8D BF 1E CA 10 EA AD B3 37
12B8: 1E C9 06 F0 1A EE B3 1E 40
12C0: D0 15 4C DF 10 A2 00 A0 C5
12C8: 00 C8 D0 FD E8 D0 F8 CE 64
12D0: BB 1E D0 03 4C 1B 10 4C DF
12D8: 2D 10 20 2F 14 20 2F 13 21
12E0: 4C F2 16 00 2D 40 2D 00 6F
12E8: D3 C0 D3 C0 D3 00 2D 40 E6
12F0: 2D 00 D3 20 FD 12 24 51 F9
12F8: 10 02 E6 50 60 85 4E 84 C1
1300: 4F A9 00 85 50 24 4E 10 50
1308: 05 38 A9 00 E5 4F 85 51 BC
1310: A2 08 06 51 26 50 06 4E 2C
1318: 90 11 18 A5 51 65 4F 85 6C
1320: 51 90 02 E6 50 24 4F 10 83
1328: 02 C6 50 CA D0 E4 60 A0 33
1330: 03 B1 FA D0 01 60 A0 04 7F
1338: B1 FA 8D A2 1E 0A 18 69 84
1340: 19 A8 B1 FA 85 FC C8 B1 66
1348: FA 85 FD A0 00 B1 FA 18 EB
1350: 71 FC 8D 9D 1E C8 B1 FA 6C
1358: 18 71 FC C9 07 90 05 EE 96
1360: 9D 1E E9 07 8D 9E 1E C8 76
1368: B1 FA 18 71 FC 8D 9F 1E BB
1370: AD 9E 1E 0A 0A 18 69 05 02
1378: A8 B1 FC 85 1C C8 B1 FC BB
1380: 85 1D C8 B1 FC 85 1E C8 E7
1388: B1 FC 85 1F A9 0A AC 98 D0
1390: 1E F0 03 18 69 07 A8 A9 46
1398: 01 91 FA C8 AD 9D 1E 91 41
13A0: FA C8 AD 9E 1E 91 FA C8 0C
13A8: AD 9F 1E 91 FA C8 AD A2 63
13B0: 1E 91 FA C8 B1 FA 85 EE A9
13B8: C8 B1 FA 85 EF A0 03 B1 21
13C0: FC 8D A0 1E C8 B1 FC 8D 53
13C8: A1 1E A9 00 8D 88 1E 8D 95
13D0: BA 1E AD 9F 1E 8D A3 1E 18
13D8: AD A1 1E 8D A4 1E 20 0F C7
13E0: 17 AC A0 1E 88 B1 FE 91 4E
13E8: EE 31 1E D1 1E F0 0A EE 6B

```


13F0: B8 1E C9 00 30 03 EE BA 5A
13F8: 1E 51 FE 11 1C 91 FE 88 21
1400: 10 E3 18 A5 EE 6D A0 1E 13
1408: 85 EE 90 02 E6 EF 18 A5 AD
1410: 1C 6D A0 1E 85 1C 90 02 57
1418: E6 1D 18 A5 1E 6D A0 1E 5E
1420: 85 1E 90 02 E6 1F EE A3 F9
1428: 1E CE A4 1E D0 B0 60 A9 3D
1430: 0A AC 98 1E F0 03 18 69 AA
1438: 07 A8 B1 FA D0 01 60 A9 E8
1440: 00 91 FA C8 B1 FA 8D 9D EA
1448: 1E C8 B1 FA 8D 9E 1E C8 83
1450: B1 FA 8D 9F 1E C8 B1 FA 2E
1458: 8D A2 1E C8 B1 FA 85 EE B3
1460: C8 B1 FA 85 EF AD A2 1E AA
1468: 0A 18 69 19 A8 B1 FA 85 E1
1470: FC C8 B1 FA 85 FD A0 03 97
1478: B1 FC 8D A0 1E C8 B1 FC E8
1480: 8D A1 1E AD 9F 1E 8D A3 AA
1488: 1E AD A1 1E 8D A4 1E 20 9C
1490: 0F 17 AC A0 1E 88 B1 EE 0B
1498: 91 FE 88 10 F9 18 A5 EE C5
14A0: 6D A0 1E 85 EE 90 02 E6 68
14A8: EF EE A3 1E CE A4 1E D0 F0
14B0: DE 60 A9 FF 85 1C A9 20 A5
14B8: 85 E6 8D 97 1E 20 F6 F3 DB
14C0: A9 00 8D A3 1E 8D 9D 1E 2A
14C8: 20 0F 17 A9 80 A0 27 91 A8
14D0: FE 88 10 FB EE A3 1E 20 BE
14D8: 0F 17 A9 80 A0 00 91 FE B2
14E0: A0 27 91 FE EE A3 1E AD 35
14E8: A3 1E C9 B8 D0 E9 20 0F AC
14F0: 17 A9 80 A0 27 91 FE 88 2F
14F8: 10 FB EE A3 1E AD A3 1E 4D
1500: C9 C0 D0 EA A9 00 8D 9D 0E
1508: 1E A9 F8 8D 9F 1E AD B3 28
1510: 1E 0A AA BD A8 16 85 1C C1
1518: BD A9 16 85 1D A0 00 8C 9E
1520: B7 1E AC B7 1E B1 1C F0 9F
1528: 66 EE B7 1E 48 8D B4 1E 1A
1530: 29 1F 8D B5 1E 68 29 60 09
1538: 8D B6 1E AD B6 1E F0 12 97
1540: C9 40 F0 1A 90 06 EE 9D 37
1548: 1E 4C 67 15 CE 9D 1E 4C 48
1550: 67 15 38 AD 9F 1E E9 08 A6
1558: 8D 9F 1E 4C 67 15 18 AD 27
1560: 9F 1E 69 08 8D 9F 1E 2C E2
1568: B4 1E 10 1B AD 9F 1E 8D DD
1570: A3 1E A9 80 A2 00 20 0F 95
1578: 17 A9 80 81 FE EE A3 1E D9

```

1580: A9 07 2C A3 1E D0 EF CE E3
1588: 85 1E D0 AF 4C 22 15 AC EB
1590: B7 1E C8 B1 1C 8D 9D 1E C2
1598: C8 B1 1C 8D A3 1E A9 B6 8F
15A0: 85 1C A9 16 85 1D A9 14 33
15A8: 8D A4 1E 20 0F 17 A0 02 A0
15B0: B1 1C 91 FE 88 10 F9 EE 44
15B8: A3 1E 18 A5 1C 69 03 85 AB
15C0: 1C 90 02 E6 1D CE A4 1E 57
15C8: D0 E1 A9 00 85 1C 85 FE AF
15D0: A9 20 85 1D A9 40 85 FF B3
15D8: A0 00 B1 1C 91 FE C8 D0 36
15E0: F9 E6 1D E6 FF A5 FF C9 34
15E8: 60 D0 EF 60 4B FF 86 CD B0
15F0: 00 07 7E 6A D1 A3 E6 66 A4
15F8: C6 90 A3 E6 65 86 C5 E3 42
1600: CC A3 E6 00 1D 0E 45 EC F0
1608: 47 25 EE 8B 6A 43 CD AA 5E
1610: C2 02 B4 45 65 00 05 96 D4
1618: 68 C3 45 A7 EB C3 83 E4 E6
1620: 83 C3 E6 C3 83 E2 A1 04 07
1628: 83 47 E6 83 C3 E3 C3 45 77
1630: E6 AC 83 C6 83 AA 83 C6 6C
1638: 83 AA 83 C6 45 65 83 00 74
1640: 1D 88 66 C6 A1 E3 A2 C8 01
1648: E1 27 E1 A1 66 C4 45 24 7A
1650: E1 63 05 EA C2 82 EA 81 0D
1658: C1 E5 46 81 05 E3 06 E4 68
1660: A1 C2 82 AD 81 C1 A6 C2 5C
1668: 82 A4 84 E1 A1 82 A1 E1 E9
1670: 81 68 05 C3 A1 E1 C3 EA 5B
1678: C1 81 E2 06 C1 00 19 0C EE
1680: 65 D3 E9 64 E2 81 C1 EE 67
1688: 8E A2 24 AA 24 A4 C9 E8 04
1690: 64 EC 00 0D 3F 46 F2 82 76
1698: E1 C2 EC C5 E2 C1 A2 C4 88
16A0: AC C2 A1 82 B1 00 05 94 5C
16A8: EC 15 95 16 F3 15 80 16 AF
16B0: 06 16 42 16 18 16 D5 AA 7E
16B8: D5 D5 AA D5 D5 AA 00 D5 27
16C0: AA 00 D5 AA 00 D5 AA 00 54
16C8: D5 AA 00 D5 AA 00 D5 AA 93
16D0: 00 D5 AA 00 D5 AA 00 D5 F6
16D8: AA 00 D5 AA 00 D5 AA 00 6C
16E0: D5 AA 00 D5 AA 00 D5 AA AB
16E8: 00 D5 AA 00 D5 AA D5 D5 BA
16F0: AA D5 AD 97 1E C9 40 A9 59
16F8: 00 2A AA BD 54 C0 8A 49 E4
1700: 01 8D 98 1E AD 97 1E 85 94
1708: E6 49 60 8D 97 1E 60 AD 84

```

1710: A3 1E 29 3F A8 B9 38 17 64
1718: 0D 97 1E 85 FF AD A3 1E EA
1720: 29 08 F0 02 A9 80 18 2C CE
1728: A3 1E 70 04 10 04 69 28 89
1730: 69 28 6D 9D 1E 85 FE 60 0A
1738: 00 04 08 0C 10 14 18 1C 46
1740: 00 04 08 0C 10 14 18 1C 4E
1748: 01 05 09 0D 11 15 19 1D 56
1750: 01 05 09 0D 11 15 19 1D 5E
1758: 02 06 0A 0E 12 16 1A 1E 66
1760: 02 06 0A 0E 12 16 1A 1E 6E
1768: 03 07 0B 0F 13 17 1B 1F 76
1770: 03 07 0B 0F 13 17 1B 1F 7E
1778: A9 00 A0 0A 91 FA A0 11 FA
1780: 91 FA A0 0F A5 EC 91 FA 3A
1788: C8 A5 ED 91 FA A0 09 B1 79
1790: FA 18 65 EC 85 EC 90 02 C0
1798: E6 ED A0 16 A5 EC 91 FA 2A
17A0: C8 A5 ED 91 FA A0 09 B1 91
17A8: FA 18 65 EC 85 EC 90 02 D8
17B0: E6 ED 60 A9 00 8D A2 1E 0E
17B8: AD A2 1E 0A 69 19 A8 B1 7D
17C0: FA 85 FC C8 B1 FA 85 FD 7C
17C8: 20 D8 17 EE A2 1E AD A2 9A
17D0: 1E A0 18 D1 FA D0 E1 60 95
17D8: A9 01 8D 9E 1E A0 03 B1 E2
17E0: FC 8D A0 1E C8 B1 FC 8D 7B
17E8: A1 1E A9 09 8D 99 1E A0 E4
17F0: 05 B1 FC 85 1C C8 B1 FC 6A
17F8: 85 1D AC 99 1E A5 EC 91 53
1800: FC C8 A5 ED 91 FC C8 8C 13
1808: 99 1E 20 36 18 A0 07 B1 F6
1810: FC 85 1C C8 B1 FC 85 1D D9
1818: AC 99 1E A5 EC 91 FC C8 93
1820: A5 ED 91 FC C8 8C 99 1E 6A
1828: 20 36 18 EE 9E 1E AD 9E 4F
1830: 1E C9 07 D0 BA 60 AD A1 24
1838: 1E 8D A4 1E A9 00 8D 9A 54
1840: 1E A0 00 A9 00 8D 9B 1E CD
1848: B1 1C 8D 9C 1E 0A AE 9E E8
1850: 1E 0A 2E 9B 1E CA D0 F9 49
1858: 2C 9C 1E 10 02 38 24 18 DB
1860: 6A 0D 9A 1E 91 EC AD 9B 75
1868: 1E 8D 9A 1E C8 CC A0 1E 19
1870: D0 D1 18 A5 1C 6D A0 1E D0
1878: 85 1C 90 02 E6 1D 18 A5 26
1880: EC 6D A0 1E 85 EC 90 02 7B
1888: E6 ED CE A4 1E D0 AD 60 BB
1890: A2 00 20 1E FB 98 60 2C 27
1898: 54 C0 2C 51 C0 20 58 FC F1

18A0: 2C C4 1E 30 3E A9 04 85 05
 18A8: 25 20 22 FC A9 0F 85 24 40
 18B0: A2 00 BD 0E 1A F0 06 20 8B
 18B8: ED FD E8 D0 F5 AD BF 1E 8D
 18C0: C9 0A 90 07 EE 17 06 E9 A4
 18C8: 0A B0 F5 69 B0 8D 18 06 71
 18D0: AD C0 1E C9 0A 90 07 EE F7
 18D8: 15 06 E9 0A B0 F5 69 B0 D3
 18E0: 8D 16 06 A9 00 8D C4 1E 96
 18E8: 20 2A 19 A9 00 8D B3 1E 2D
 18F0: AD 61 C0 30 05 8D A5 1E 33
 18F8: 10 1D 2C A5 1E 30 18 8D C7
 1900: A5 1E EE B3 1E AD B3 1E D2
 1908: C9 07 D0 05 A9 00 8D B3 67
 1910: 1E 18 69 B1 8D BF 04 A0 B3
 1918: 00 A2 00 CA D0 FD 88 D0 FF
 1920: FA AD 00 C0 10 CA 2C 10 5B
 1928: C0 60 A2 00 BD 4E 19 F0 71
 1930: 1C 85 25 20 22 FC E8 BD 0D
 1938: 4E 19 85 24 E8 BD 4E 19 BE
 1940: F0 07 E8 20 ED FD 4C 3D 09
 1948: 19 E8 4C 2C 19 60 07 10 F5
 1950: CD C9 C1 CD C9 A0 C9 C3 19
 1958: C5 00 09 11 CC C5 D6 C5 90
 1960: CC A0 B1 00 0C 09 D0 D2 50
 1968: C5 D3 D3 A0 C1 CE D9 A0 94
 1970: CB C5 D9 A0 D4 CF A0 C2 29
 1978: C5 C7 C9 CE 00 00 2C 54 52
 1980: C0 2C 51 C0 20 58 FC A2 53
 1988: 00 BD FD 19 F0 1A 85 25 9B
 1990: 20 22 FC E8 BD FD 19 85 27
 1998: 24 E8 BD FD 19 F0 06 20 67
 19A0: ED FD E8 D0 F5 E8 D0 E1 4A
 19A8: AD B3 1E 18 69 B1 8D 96 A7
 19B0: 07 18 AD B3 1E 69 B0 8D 63
 19B8: BE 04 AD BF 1E C9 0A 90 B9
 19C0: 07 EE BF 05 E9 0A B0 F5 49
 19C8: 69 B0 8D C0 05 AD C0 1E 18
 19D0: C9 0A 90 07 EE BD 05 E9 4F
 19D8: 0A B0 F5 69 B0 8D BE 05 CF
 19E0: A9 02 8D BC 1E A9 00 8D 0B
 19E8: BD 1E 8D BE 1E CE BE 1E E6
 19F0: D0 FB CE BD 1E D0 F6 CE 31
 19F8: BC 1E D0 F1 60 07 10 CC 56
 1A00: C5 D6 C5 CC 00 09 11 C3 5C
 1A08: C1 D2 D3 00 0B 0F D3 C3 4C
 1A10: CF D2 C5 A0 B0 B0 B0 FD
 1A18: 00 00 20 20 20 53 20 20 01
 1A20: 32 45 20 40 20 20 20 20 A8
 1A28: 20 00 20 20 20 20 20 7F B3

1A30: 20 20 08 43 1A 64 1A 85 CD
1A38: 1A A6 1A C7 1A E8 1A 09 94
1A40: 1B 2A 1B 01 03 00 03 11 2F
1A48: 4B 1B 7E 1B 20 35 20 20 A0
1A50: 20 20 20 20 20 20 20 46 AA
1A58: 2E 44 20 20 20 20 41 20 DE
1A60: 20 20 20 20 01 01 01 04 C4
1A68: 10 97 1C D7 1C 20 20 20 4D
1A70: 20 20 20 20 20 20 20 20 A4
1A78: 20 20 3A 20 20 20 20 20 EF
1A80: 20 4F 20 0F 23 01 04 08 BA
1A88: 04 08 17 1C 37 1C 20 30 FF
1A90: 00 20 30 20 20 36 20 20 0F
1A98: 20 20 49 20 20 45 20 20 86
1AA0: 20 20 20 20 20 20 01 01 77
1AA8: 06 04 10 17 1D 57 1D 34 09
1AB0: 20 31 20 20 20 2E 20 53 94
1AB8: 45 20 20 20 20 20 20 20 7F
1AC0: 20 20 20 20 20 20 00 01 95
1AC8: 03 06 03 11 B1 1B E4 1B 50
1AD0: 20 20 20 20 4F 20 55 20 E8
1AD8: 20 20 20 20 20 20 31 2E 3D
1AE0: 30 31 20 20 00 46 00 20 B8
1AE8: 00 06 06 04 10 17 1E 57 10
1AF0: 1E 00 20 20 20 20 3B 20 52
1AF8: 20 20 20 2E 7F 20 20 20 09
1B00: 24 20 20 20 7F 20 20 20 33
1B08: 20 00 06 08 04 08 57 1C 9A
1B10: 77 1C 20 20 20 41 20 44 99
1B18: 20 20 20 20 20 20 20 52 80
1B20: 20 20 7F 20 20 4F 20 20 FE
1B28: 20 20 00 06 01 04 10 97 A6
1B30: 1D D7 1D 7F 20 22 20 20 70
1B38: 20 20 20 20 20 20 20 20 6E
1B40: 20 20 20 20 46 20 20 20 A7
1B48: 49 20 20 D2 84 80 D5 8A B8
1B50: 80 D5 8A 80 D5 8A 80 D5 45
1B58: 8A 80 D5 8A 80 D5 8A 80 48
1B60: FD 8B 80 9F 8F 80 81 88 8C
1B68: 80 81 88 80 81 88 80 85 0D
1B70: 8A 80 D5 8A 80 D5 8A 80 60
1B78: D5 8A 80 89 89 80 FE 87 B8
1B80: 80 FF 8F 80 FF 8F 80 FF 30
1B88: 8F 80 FF 8F 80 FF 8F 80 43
1B90: FF 8F 80 FF 8F 80 FF 8F C8
1B98: 80 FF 8F 80 FF 8F 80 FF 48
1BA0: 8F 80 FF 8F 80 FF 8F 80 5B
1BA8: FF 8F 80 FF 8F 80 FF 8F E0
1BB0: 80 89 89 80 D5 8A 80 D5 72

```

1BB8: 8A 80 D5 8A 80 85 8A 80 67
1BC0: 81 88 80 81 88 80 81 88 D3
1BC8: 80 9F 8F 80 FD 8B 80 D5 16
1BD0: 8A 80 D5 8A 80 D5 8A 80 C0
1BD8: D5 8A 80 D5 8A 80 D5 8A 96
1BE0: 80 D3 84 80 FF 8F 80 FF 24
1BE8: 8F 80 FF 8F 80 FF 8F 80 A3
1BF0: FF 8F 80 FF 8F 80 FF 8F 29
1BF8: 80 FF 8F 80 FF 8F 80 FF A8
1C00: 8F 80 FF 8F 80 FF 8F 80 BC
1C08: FF 8F 80 FF 8F 80 FF 8F 42
1C10: 80 FF 8F 80 FE 87 80 A9 43
1C18: DD EA 80 AD F1 8A 81 AD 4F
1C20: E0 AA 81 A9 E0 AA 81 A9 9C
1C28: E0 AA 81 AD E0 AA 81 AD E8
1C30: F1 8A 81 A9 DD EA 80 FF 6A
1C38: FF FF 80 FF FF FF 81 FF 83
1C40: FF FF 81 FF FF FF 81 FF AB
1C48: FF FF 81 FF FF FF 81 FF B3
1C50: FF FF 81 FF FF FF 80 D6 90
1C58: BA 95 81 D1 8E B5 81 D5 C4
1C60: 86 B4 81 D5 86 94 81 D5 F5
1C68: 86 94 81 D5 86 B4 81 D1 72
1C70: 8E B5 81 D6 BA 95 81 FE 29
1C78: FF FF 81 FF FF FF 81 FF E3
1C80: FF FF 81 FF FF FF 81 FF EB
1C88: FF FF 81 FF FF FF 81 FF F3
1C90: FF FF 81 FE FF FF 81 00 EB
1C98: A0 85 80 00 A8 94 80 00 2B
1CA0: A8 95 80 00 AA D5 80 00 50
1CA8: AA C5 81 C0 BE D5 80 C0 F2
1CB0: B0 95 80 D0 E0 95 80 90 B2
1CB8: C0 85 80 94 C0 85 80 D2 FB
1CC0: 80 81 00 CA A0 81 00 D4 26
1CC8: AA 80 00 C8 AA 80 00 90 EA
1CD0: 8A 80 00 A0 8D 80 00 00 E6
1CD8: E0 87 80 00 F8 9F 80 00 BA
1CE0: F8 9F 80 00 FE FF 80 00 86
1CE8: FE FF 81 C0 FF FF 80 C0 9E
1CF0: FF 9F 80 F0 FF 9F 80 F0 A0
1CF8: FF 87 80 FC FF 87 80 FE 11
1D00: FF 81 00 FE FF 81 00 FC 8D
1D08: BF 80 00 F8 BF 80 00 F0 C2
1D10: 8F 80 00 E0 8F 80 00 A0 5F
1D18: 8D 80 00 90 8A 80 00 C8 61
1D20: AA 80 00 D4 AA 80 00 CA 3F
1D28: A0 81 00 D2 80 81 00 94 DE
1D30: C0 85 80 90 C0 85 80 D0 33
1D38: E0 95 80 C0 B0 95 80 C0 02
1D40: BE D5 80 00 AA C5 81 00 CE

```


1D48: AA D5 80 00 AB 95 80 00 F9
1D50: A8 94 80 00 A0 85 80 E0 11
1D58: 8F 80 00 F0 8F 80 00 F8 01
1D60: BF 80 00 FC BF 80 00 FE 69
1D68: FF 81 00 FE FF 81 00 FC F5
1D70: FF 87 80 F0 FF 87 80 F0 BB
1D78: FF 9F 80 C0 FF 9F 80 C0 F6
1D80: FF FF 80 00 FE FF 81 00 C5
1D88: FE FF 80 00 F8 9F 80 00 99
1D90: F8 9F 80 00 E0 87 80 D0 36
1D98: 82 80 00 94 8A 80 00 D4 A8
1DA0: 8A 80 00 D5 AA 80 00 D1 C6
1DA8: AA 80 00 D5 BE 91 00 D4 86
1DB0: 86 81 00 D4 83 85 80 D0 DF
1DB8: 81 84 80 D0 81 94 80 C0 12
1DC0: 80 A5 80 C0 82 A9 80 00 7C
1DC8: AA 95 80 00 AA 89 80 00 4A
1DD0: A8 84 80 00 D0 82 80 F0 13
1DD8: 83 80 00 FC 8F 80 00 FC 40
1DE0: 8F 80 00 FF BF 80 00 FF 03
1DE8: BF 80 00 FF FF 81 00 FC 26
1DF0: FF 81 00 FC FF 87 80 F0 6B
1DF8: FF 87 80 F0 FF 9F 80 C0 74
1E00: FF BF 80 C0 FF BF 80 00 48
1E08: FE 9F 80 00 FE 8F 80 00 F2
1E10: F8 87 80 00 F0 83 80 00 51
1E18: D0 82 80 00 A8 84 80 00 C5
1E20: AA 89 80 00 AA 95 80 C0 91
1E28: 82 A9 80 C0 80 A5 80 D0 98
1E30: 81 94 80 D0 81 84 80 D4 63
1E38: 83 85 80 D4 86 81 00 D5 05
1E40: BE 81 00 D1 AA 80 00 D5 86
1E48: AA 80 00 D4 8A 80 00 94 32
1E50: 8A 80 00 D0 82 80 00 00 15
1E58: F0 83 80 00 F8 87 80 00 E4
1E60: FE 8F 80 00 FE 9F 80 C0 48
1E68: FF BF 80 C0 FF BF 80 F0 A1
1E70: FF 9F 80 F0 FF 87 80 FC CF
1E78: FF 87 80 FC FF 81 00 FF 7C
1E80: FF 81 00 FF BF 80 00 FF 1D
1E88: BF 80 00 FC 8F 80 00 FC 10
1E90: 8F 80 00 F0 83 80 00 00 E1

2

Education



Hickory, Dickory, Dock

Barbara H. Schulak

Apple version by Tim Victor

This fun, educational program helps children learn the concepts of telling time by relating a digital clock display to a conventional clock face. For all Apple II series computers using either DOS 3.3 or ProDOS.

"Hickory, Dickory, Dock" offers an enjoyable way for children to learn to tell time. Type in the program and save a copy before running it.

When you run Hickory, Dickory, Dock, it displays a round clock face as well as a digital display. Four different activities are available. The first option lets youngsters practice telling time. As the positions of the clock hands change on the screen, the digital clock display changes as well. This shows the relationship between the spatial position of hands on a clock face and the numeric representation of time.

The other three activities test a youngster's time-telling ability for hours only, hours and half-hours, or five-minute intervals.

To move the hands to the correct position, press the 1 key to shift the minutes hand and the 2 key to move the hours hand. Press Return when the hands are in the right places. After three incorrect choices, the program automatically moves the clock hands to the correct position.

Hit the Esc (Escape) key to return to the main menu.

Hickory, Dickory, Dock

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
6F 100 PI = 4 * ATN (1)
ED 110 DIM DP(4,7),CX(12),CY(12)
77 120 GOSUB 840: GOSUB 960: GOSUB 1120: GOSUB 1160:
      POKE 6,0: POKE 7,138
25 130 IF PEEK (190 * 256) = 76 THEN PRINT CHR$(4);
      "PR#A$300": GOTO 150
BA 140 POKE 54,0: POKE 55,3: CALL 1002
32 150 HGR : GOSUB 510
```

```

39 160 HGR : HOME : GOSUB 600
1E 170 VTAB 19: HTAB 16: PRINT "1": HTAB 16: PRINT "
2"
1C 180 FOR I = 0 TO 1: HCOLOR= 5 + I: FOR J = 0 TO 1
: HPLLOT 18,147 + J + 8 * I TO 52 + 8 * I,147
+ J + 8 * I: NEXT : NEXT
85 190 VTAB 21: HTAB 3: PRINT "TO MOVE";: HTAB 14: P
RINT "PRESS": HTAB 2: PRINT "THIS HAND";: HTA
B 13: PRINT "THIS KEY"
BB 200 CH = 213: FOR CV = 138 TO 150 STEP 12: GOSUB
770: NEXT
3C 210 FOR I = 1 TO 4: DD(I) = 10: FOR J = 1 TO 7: DP(
I,J) = 0: NEXT : NEXT
E5 220 HH = 84:HV = 35:MH = 83:MV = 24
D7 230 IF GM = 1 THEN 410
46 240 NW = 0:TH = INT ( RND (1) * 12) + 1:TM = INT
( RND (1) * 60)
FA 242 IF RH = TH THEN 240
F3 245 RH = TH:RM = TM
1F 250 IF GM = 2 THEN TM = 0
B7 260 IF GM = 3 THEN TM = 30 * INT (TM / 30)
5D 270 IF GM = 4 THEN TM = 5 * INT (TM / 5)
77 280 HR = TH:MN = TM: GOSUB 610
C5 290 HR = 1:MN = 0
4C 300 VTAB 24: HTAB 2: PRINT "PRESS RETURN TO ANSWE
R, ESC FOR MENU";
4F 310 GOSUB 670
CF 320 GOSUB 460: IF A$ = CHR$ (27) THEN 150
33 330 IF A$ = CHR$ (13) THEN 350
96 340 GOTO 310
23 350 IF TH = HR AND TM = MN THEN 390
47 360 NW = NW + 1: IF NW < 3 THEN VTAB 24: HTAB 2:
PRINT SPC( 38);: HTAB 5: PRINT "THAT IS NOT C
ORRECT, TRY AGAIN";: FOR I = 1 TO 1000: NEXT
: GOTO 300
C8 370 HR = TH:MN = TM: GOSUB 670: VTAB 24: HTAB 2:
PRINT SPC( 38);: HTAB 7: PRINT "THIS IS THE C
ORRECT ANSWER";
DE 380 FOR I = 1 TO 1500: NEXT : GOTO 240
37 390 VTAB 24: HTAB 2: PRINT SPC( 38);: HTAB 10: PR
INT "CORRECT! GOOD ANSWER";
7F 400 FOR I = 1 TO 1000: NEXT : GOTO 240
DD 410 VTAB 24: HTAB 11: PRINT "PRESS ESC FOR MENU";
B9 420 HR = 1:MN = 0
09 430 GOSUB 610: GOSUB 670
D4 440 GOSUB 460: IF A$ = CHR$ (27) THEN 150
1C 450 GOTO 430
2A 460 VTAB 24: HTAB 1: GET A$: IF A$ = CHR$ (27) TH
EN RETURN
94 470 IF A$ = CHR$ (13) THEN RETURN
0E 480 IF A$ = "1" THEN HR = HR + 1 - 12 * (HR = 12)
: RETURN

```

```

EF 490 IF A$ = "2" THEN MN = MN + 5 - 60 * (MN = 55)
      : RETURN
16 500 GOTO 460
56 510 TEXT : HOME : VTAB 6: HTAB 8: PRINT "PRESS KE
      Y TO CHOOSE GAME:"
68 520 VTAB 10: HTAB 7: PRINT "1: PRACTICE"
8A 530 VTAB 12: HTAB 7: PRINT "2: HOURS TEST"
2D 540 VTAB 14: HTAB 7: PRINT "3: HOURS AND HALF HOU
      RS TEST"
0B 550 VTAB 16: HTAB 7: PRINT "4: FIVE MINUTE INTERV
      ALS TEST"
59 560 VTAB 18: HTAB 7: PRINT "5: QUIT"
26 570 VTAB 24: GET A$: IF A$ < "1" OR A$ > "5" THEN
      570
58 580 IF A$ = "5" THEN END
6B 590 GM = VAL (A$): RETURN
69 600 FOR I = 1 TO 12: VTAB CY(I): HTAB CX(I): PRIN
      T I: NEXT : RETURN
CE 610 DE = 1: HC = 160: DC = INT (HR / 10): IF DC = 0
      THEN DC = 10
AC 620 IF DC < > DD(1) THEN GOSUB 790
45 630 DD(1) = DC: HC = 184: DE = 2: DC = HR - 10 * INT
      (HR / 10): IF DC < > DD(2) THEN GOSUB 790
6F 640 DD(2) = DC: HC = 220: DE = 3: DC = INT (MN / 10)
      : IF DC < > DD(3) THEN GOSUB 790
04 650 DD(3) = DC: HC = 246: DE = 4: DC = MN - DC * 10:
      IF DC < > DD(4) THEN GOSUB 790
13 660 DD(4) = DC: RETURN
95 670 HCOLOR= 0: GOSUB 700: GOSUB 720
A7 680 GOSUB 690: GOSUB 710: RETURN
E3 690 A = (HR / 6 + MN / 360) * PI: HV = 68 - 33 * C
      OS (A): HH = 84 + 44 * SIN (A): HCOLOR= 5
F6 700 H PLOT 84,68 TO HH,HV: H PLOT 83,67 TO HH - 1,H
      V - 1: RETURN
94 710 A = MN / 30 * PI: MV = 68 - 44 * COS (A): MH =
      83 + 59 * SIN (A): HCOLOR= 6
DC 720 H PLOT 84,68 TO MH,MV: H PLOT 83,67 TO MH - 1,M
      V - 1: RETURN
65 730 FOR I = - 2 TO 2: H PLOT CH + I,CV - 4 + ( ABS
      (I) = 2) TO CH + I,CV + 4 - ( ABS (I) = 2):
      NEXT
1F 740 RETURN
5B 750 FOR I = - 1 TO 1: H PLOT CH - 5 - (I = 0),CV +
      I TO CH + 5 + (I = 0),CV + I: NEXT
23 760 RETURN
27 770 HCOLOR= 2: FOR I = - 2 TO 2: H PLOT CH + I,CV
      - 2 + ( ABS (I) < 2) TO CH + I,CV + 2 - ( ABS
      (I) < 2): NEXT
27 780 RETURN
0B 790 CI = 0: CH = HC + 10: FOR CV = 132 TO 156 STEP

```



```

12: GOSUB 830: IF DP(DE,CI) < > PC THEN GOSU
B 750:DP(DE,CI) = PC
FF 800 NEXT
7A 810 FOR CV = 138 TO 150 STEP 12: FOR CH = HC + 2
TO HC + 18 STEP 16: GOSUB 830: IF DP(DE,CI) <
> PC THEN GOSUB 730:DP(DE,CI) = PC
5E 820 NEXT : NEXT : RETURN
CB 830 CI = CI + 1:PC = VAL ( MID$ (SS$(DC),CI,1)):
HCOLOR= 2 * PC: RETURN
F8 840 FOR I = 35456 TO I + 72 STEP 8: POKE I,128: P
OKE I + 7,128
DB 850 FOR J = 1 TO 6: READ A: POKE I + J,A: NEXT :
NEXT : RETURN
90 860 DATA 188,230,246,238,230,188
B1 870 DATA 152,156,152,152,152,188
75 880 DATA 188,230,176,140,230,254
88 890 DATA 188,230,176,224,230,188
9F 900 DATA 176,184,180,254,176,176
5D 910 DATA 254,134,190,224,230,188
79 920 DATA 188,134,190,230,230,188
8A 930 DATA 254,224,176,152,140,140
14 940 DATA 188,230,188,230,230,188
80 950 DATA 188,230,230,252,176,152
EB 960 FOR I = 768 TO I + 87: READ A: POKE I,A: NEXT
: RETURN
A4 970 DATA 216,120,133,69,134,70
50 980 DATA 132,71,166,7,10,10
5B 990 DATA 176,4,16,62,48,4
A2 1000 DATA 16,1,232,232,10,134
4A 1010 DATA 27,24,101,6,133,26
8D 1020 DATA 144,2,230,27,165,40
7F 1030 DATA 133,8,165,41,41,3
6B 1040 DATA 5,230,133,9,162,8
28 1050 DATA 160,0,177,26,36,50
73 1060 DATA 48,2,73,127,164,36
5F 1070 DATA 145,8,230,26,208,2
A7 1080 DATA 230,27,165,9,24,105
1F 1090 DATA 4,133,9,202,208,226
71 1100 DATA 165,69,166,70,164,71
5C 1110 DATA 88,76,240,253
A1 1120 FOR DC = 0 TO 10: READ SS$(DC): NEXT : RETUR
N
02 1130 DATA 1011111,0000101,1110110,1110101
14 1140 DATA 0101101,1111001,1111011,1000101
8E 1150 DATA 1111111,1111101,0000000
42 1160 FOR I = 1 TO 12: READ CY(I),CX(I): NEXT : RE
TURN
AC 1170 DATA 2,18,5,22,9,23
23 1180 DATA 13,22,16,18,17,12
5B 1190 DATA 16,6,13,2,9,1
65 1200 DATA 5,2,2,6,1,12

```

Skyscape

Robert M. Simons

Apple version by Tim Victor

This unique program, written by a planetarium director, presents the sky as it can be viewed at any date and time from the year 1977 onward—including zodiac constellations and all the visible planets. It also calculates planet tables, positions of the sun, and phases of the moon for any date and time from 1977 into the future. And if you missed Halley's Comet, "Skyscape" can show you where it was as it neared Earth in late 1985 and early 1986. Skyscape is both educational and entertaining. For Apple II series computers using either DOS 3.3 or ProDOS.

For thousands of years the sun, moon, and planets in our solar system have excited human imagination. In ancient times they were regarded as gods whose distant motions influenced the course of earthly events. Though we now understand more about the true nature of celestial objects, many facts remain unknown, and a brilliant nighttime sky still presents an inspiring spectacle.

Whether you're seriously interested in the sky or just casually curious, "Skyscape" is a convenient tool for extending your knowledge. It opens a movable window on the heavens, displaying the position of our sun, moon, and neighboring planets from almost any location on Earth, at any point in time from 1977 into the distant future. Since it performs all the necessary calculations, you can enjoy and learn from this program even if you're not an expert in astronomy. In addition to providing data about the position of celestial objects, it draws a sky map on the screen, showing each object as it would appear to you at the chosen location and time.

To get started, type in Skyscape and save a copy before running it.

Past, Present, or Future

Skyscape begins by asking you to answer several questions. Enter the year, choosing any year from 1977 forward. In some ways this is the most important input of all, since objects in our solar system move significantly from one year to the next. After you choose the year, Skyscape allows you to enter the month and day.

Next, you must enter the latitude (north/south position on Earth) from which you wish to view the sky. Latitude 0 places you, the observer, at the equator. Latitudes 1–90 place you in the northern hemisphere (north of the equator). To choose a southern latitude (south of the equator), enter a negative number from -1 to -90 . Skyscape generally represents southerly locations with negative values.

To get you started, here's a short list of some cities, and the latitude values you'd enter in Skyscape (you can easily find your location's latitude by calling the closest National Weather Service office). The actual latitudes of these cities have been rounded to the closest whole number.

City	Latitude
New York	41
Los Angeles	34
Anchorage	61
London	51
Paris	49
Athens	38
Johannesburg	-26
Peking	39
Sydney	-34
Rio de Janeiro	-23

Whenever Skyscape asks for information, it checks your entry to make sure it's in the acceptable range. If you enter an illegal value, the program displays an error message and gives you another chance.

The Sun and Moon

Though very different in size and composition, the sun and moon are alike in being the largest celestial objects visible from Earth. After you enter the date and latitude, Skyscape displays a table of data for the sun and moon. In addition to the date, day of the year, and latitude north or south, you'll see the following information:

- Sun's geocentric angle. This figure represents the sun's position as a number of degrees relative to the vernal equinox. The vernal equinox is where the sun is located when spring begins in the northern hemisphere (the same time that autumn begins in the southern hemisphere).
- Sun's declination. The number of degrees north or south of the equator. Negative values indicate a southerly location.

- Sun's altitude at noon. The location of the sun in degrees from the northern or southern horizon at noon.
- Sun's right ascension. Just as longitude and latitude indicate locations on the Earth, *right ascension* and *declination* are used to pinpoint locations in the sky. For this purpose the sky is visualized as a gigantic sphere surrounding the Earth. Declination locates a point vertically in the celestial sphere, and right ascension locates it horizontally. Right ascension values are given in *hours* and *minutes* in the range 0:00–23:59. Right ascension 0:00 is exactly at the vernal equinox. Larger right ascension values lie to the east of smaller ones.
- Right ascension at 9:00 p.m. The right ascension which would be on the meridian at 9:00 p.m. This coordinate system would be found on star charts. By comparing this number with those charts, you can tell what stars and constellations would be visible at that time.
- Moon's age. The number of days since the last new moon.
- Moon's elongation. The location of the moon in degrees east or west of the sun.
- Moon's phase. The phase of the moon on this particular day.

The Planet Table

After viewing the sun and moon display, press P to continue to the next display screen, which contains the planet table. (Press D if you wish to enter a new date.) The planet table shows vital information about the visible planets (through Uranus, which is at the limit of our visibility). The table shows the position of each planet in right ascension and degrees east or west of the sun. It also shows the distance of each planet from Earth in millions of miles.

If you'd rather see the distance in kilometers, change the value of ES in line 80 of the program from 93 to 149.6.

Some planets have an asterisk to the left of the right ascension figure. This signifies that they're visible at 9:00 this evening. For reference, the planet table also includes the sun's present right ascension and its right ascension at 9:00 p.m. Press D to input a new date or S to view a graphics display of the sky at any time in the current day.

The Visible Skyscape

After selecting the sky display, you must enter the hour when you wish to view the sky. The hour value should be a whole

number 0–23 (enter 14 for 2:00 p.m., 22 for 10:00 p.m., and so on). You'll also need to enter the minutes (0–59). Skyscape then displays the time and offers you a chance to enter different values. Press Return when you're satisfied with the time.

Skyscape now displays the sky as it would appear at the chosen latitude, date, and time. Since the sky looks very different from different places on Earth, the latitude affects the display considerably. If your latitude is in the range 24–90 degrees north or south, the sky shows a dashed line representing the position of the celestial equator, along with symbols representing the sun, moon, and planets visible at that time. If your latitude is in the tropical region—from 23½ degrees north to 23½ degrees south—the dashed line indicates a position directly overhead.

If you're viewing in the northern hemisphere, north is above the dashed line and south is below it. In the southern hemisphere these directions are reversed. Below the sky display is a key that interprets the symbols used to represent celestial objects. If more than one object is positioned at the same spot, the symbols are displayed above each other.

At the bottom of the sky you may see two-letter abbreviations. These represent zodiac constellations that would be visible from your chosen vantage point. Skyscape uses the abbreviations AR (Aries), PI (Pisces), AQ (Aquarius), CP (Capricorn), SA (Sagittarius), SC (Scorpio), LI (Libra), VI (Virgo), LE (Leo), CA (Cancer), GE (Gemini), and TA (Taurus). Each constellation is located above the spot where its abbreviation appears. In northern latitudes, the border of each constellation's zone begins at its abbreviation and extends left. In southern latitudes, the constellation extends right from the position of its abbreviation.

Daytime skies are shown in white and nighttime skies in black. Skyscape doesn't calculate the actual rising or setting time of the sun. Average rising and setting times of 6:00 a.m. and 6:00 p.m. are used in every case. You may obtain exact rising and setting times from local newspapers. However, keep in mind that there's usually about an hour of twilight before sunrise and after sunset.

Halley's Comet

In addition to permanent objects, Skyscape's graphics display includes Halley's Comet, which was visible during late 1985

and early 1986. If you missed the comet when it actually appeared (not that difficult to do, really), you can see where it was *supposed* to appear in the skies.

Choose a date from November 1, 1985, to May 29, 1986, and Skyscape calculates the position of Halley's Comet and includes it in the graphics display (if it would have been visible at the place and time you select). The comet's position is based on the best predictions available when this article was first written (summer 1985).

While Skyscape is generally accurate, it bases most position calculations on circular orbits. This introduces a certain element of error, since no object in our solar system has a perfectly circular orbit. The position error is most pronounced for Mercury and Mars (whose orbits are quite elliptical), but does not significantly affect other objects. I've found Skyscape accurate enough for my own purposes, which include planning astronomy classes and planetarium displays.

Skyscape

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```

10 60 GOSUB 1940
40 70 D$ = "000031059090120151181212243273304334":K1
    = 1440: DIM HC(22):MM$ = "041081040"
84 80 M$ = "286317345011041072102133164194225255":D$
    (1) = "S":D$(2) = "N":ES = 93
23 90 A$ = "JANFEBMARAPR MAYJUNJUL AUGSEP OCTNOV DEC":OO
    $ = "OUT OF RANGE!!"
2A 100 MD$ = "312831303130313130313031":D9 = ATN (1)
    / 45: READ EE: READ M9: DIM P(6,6)
C0 110 DEF FN R(X) = INT (X * 100 + .5) / 100
46 120 DEF FN S(X) = INT (X * 10 + .5) / 10
08 130 FOR Y = 1 TO 2: FOR X = 1 TO 6: READ P(X,Y):
    NEXT : NEXT :Y = 0
73 140 FOR X = 1 TO 6: READ P$(X),P(X,3): NEXT
14 150 FOR X = 1 TO 7:PP(X) = X + 85: NEXT
1F 160 J$ = "SAT SUN MON TUE WED THU FRI": FOR X = 1 TO 12
    : READ F$
8B 170 CC$ = CC$ + " " + F$: NEXT :CC$ = CC$ + C
    C$:F$ = RIGHT$(CC$,9):CC$ = F$ + CC$
C1 180 FOR X = 1 TO 8: READ PH$(X): NEXT
2E 190 FOR X = 1 TO 22: READ HC(X): NEXT :R$ = "0":T
    $ = "00": GOTO 720
3B 200 CC = MT - 720: IF CC < 0 THEN CC = CC + K1
2C 210 CC = CC / 120:CD = CC - INT (CC):CC = INT (CC
    ):CD = INT (CD * 7 + .2):CC = 81 - (CC * 7 +
    CD)

```



```

51 220 GOSUB 1770: IF LL < 0 THEN GOSUB 5000
92 225 VTAB 17: PRINT CD$; RETURN
DB 230 HOME : HTAB 10: PRINT "** DAYS SKY **": VTAB
3: GOSUB 1550: HTAB 31: PRINT R$:"T$
DF 240 VTAB 5: HTAB 1: PRINT "INPUT THE TIME:": PRIN
T "-----"
BB 245 PRINT : PRINT "      HOUR (0-23) ";: GOSUB 224
0: IF I$ < > "" THEN T1 = VAL (I$)
B3 250 IF T1 < 0 OR T1 > 23 THEN PRINT 00$: GOTO 245
A3 255 PRINT : PRINT "      MINUTE (0-59) ";: GOSUB 224
0: IF I$ < > "" THEN T2 = VAL (I$)
9D 260 IF T2 < 0 OR T2 > 59 THEN PRINT 00$: GOTO 255
85 270 R$ = STR$ (T1):T$ = STR$ (T2): IF LEN (T$) =
1 THEN T$ = "0" + T$
EB 280 VTAB 13: PRINT "TIME-- "R$:"T$
AA 290 PRINT : GOSUB 2020: IF I$ = "N" THEN 230
10 300 HOME :T3 = T1 * 60 + T2 + AA - 720: IF T3 < 0
THEN T3 = T3 + K1
20 310 IF T3 > K1 THEN T3 = T3 - K1
B7 320 MT = T3 - 360: IF MT < 0 THEN MT = MT + K1
FF 330 PT = T3 + 360: IF PT > K1 THEN PT = PT - K1
16 340 HTAB 4: GOSUB 1550: HTAB 31: PRINT R$:"T$
F7 350 TM = VAL (R$ + "." + T$): IF TM >= 6 AND TM
<= 18 THEN INVERSE
04 360 XX = 7 + LC: VTAB 3: HTAB 1: FOR X = 1 TO 14:
IF X = XX THEN GOTO 380
CE 370 PRINT SPC( 40);: GOTO 390
B6 380 PRINT "-----
";
02 390 NEXT X: NORMAL : GOSUB 200: INVERSE : IF LL <
0 THEN 395
31 393 IF LL > 24 THEN PRINT "E" SPC( 18)"S" SPC( 19
)"W": GOTO 400
30 394 PRINT "UP-NORTH" SPC( 5)"----OVERHEAD" SPC( 5
)"DOWN-SOUTH": GOTO 400
05 395 IF LL < - 24 THEN PRINT "W" SPC( 18)"N" SPC(
19)"E": GOTO 400
C0 397 PRINT "UP-SOUTH" SPC( 5)"----OVERHEAD" SPC( 5
)"DOWN-NORTH"
09 400 T4 = AA: GOSUB 610:Y8 = 888
43 410 IF Y9 = 999 THEN 450
A6 420 GOSUB 4000:Y8 = Y9: IF A1 < 0 THEN 450
06 430 IF U9 > 16 OR U9 < 3 THEN 450
80 440 VTAB U9: HTAB 40 - Y9: PRINT CHR$ (42)
97 450 T4 = AA + M2 * K1: IF T4 > K1 THEN T4 = T4 -
K1
EB 460 GOSUB 610: IF Y9 = 999 THEN 500
97 470 MM = INT (M1 / 9.83333) + 1: GOSUB 710
10 480 GOSUB 4000: IF U9 > 16 OR U9 < 3 THEN 500

```

```

60 490 VTAB U9: HTAB 40 - Y9: PRINT CHR$ (MM);: IF A
    BS (Y8 - Y9) <= .5 THEN NORMAL : HTAB 40 - Y
    9: PRINT CHR$ (81);: INVERSE
70 500 FOR X = 1 TO 7: IF X = 7 THEN 2140
60 510 T4 = P(X,6): GOSUB 610: IF Y9 = 999 THEN 560
87 520 U9 = SIN ((P(X,6) / 4) / (1 / D9)):U9 = INT (
    - 3 * U9 + .5)
13 530 GOSUB 4005: IF U9 < 3 OR U9 > 16 THEN 560
8E 540 SR = INT ((U9 - 1) / 8):Z = PEEK (1024 - SR *
    984 + (U9 - 1) * 128 + 39 - Y9): IF Z > 127
    THEN Z = Z - 128
A0 545 IF Z < > 32 AND Z < > 45 THEN U9 = U9 + 2 * (
    LL >= 0) - 1: GOTO 540
2A 550 VTAB U9: HTAB 40 - Y9: PRINT CHR$ (PP(X));
EB 560 NEXT X: NORMAL
F3 570 VTAB 20: HTAB 1: PRINT "VMERCURY   WVENUS   X
    MARS       YJUPITER"
2F 580 PRINT "ZSATURN   [URANUS   *SUN       )Q(MOON"
99 590 HTAB 3: INVERSE : PRINT "Q";: NORMAL : PRINT
    "NEW MOON + SUN   "B$
92 600 PRINT : PRINT "T- NEW TIME,P- P. TABLE,D- DAT
    E,L- LAT";: GOTO 1700
7F 610 Y9 = 999: IF MT < PT THEN 660
36 620 IF (T4 >= MT) OR (T4 <= PT) THEN 640
1C 630 RETURN
87 640 IF (T4 >= MT) AND (T4 <= K1) THEN 680
7C 650 T4 = T4 + K1: GOTO 680
C4 660 IF (T4 >= MT) AND (T4 <= PT) THEN GOTO 680
24 670 RETURN
7A 680 Y9 = INT ((T4 - MT) / 18 + .5): IF Y9 = 40 TH
    EN Y9 = 39
28 690 RETURN
6A 700 U9 = SIN ((T4 / 4) / (1 / D9)):U9 = INT ( - 3
    * U9 + .5): RETURN
56 710 MM = VAL ( MID$ (MM$,3 * MM - 2,3)): IF LL <
    0 AND MM < > 81 THEN MM = ABS (MM - 81)
2D 715 RETURN
DA 720 HOME : VTAB 2: HTAB 7: PRINT "***** SKYS
    CAPE *****": VTAB 4: PRINT "DATE INPUT"
56 730 PRINT "-----": IF Y < > 0 THEN VTAB 6: G
    OSUB 1550: PRINT : PRINT
E5 740 PRINT "YEAR   ": GOSUB 2240: IF I$ < > "" THE
    N Y = VAL (I$)
14 745 IF Y < 1977 THEN PRINT "MUST BE AFTER 1977":
    GOTO 740
D3 750 GOSUB 1600: PRINT : PRINT "MONTH (1-12) ": G
    OSUB 2240: IF I$ < > "" THEN M = VAL (I$)
B5 755 IF M < 1 OR M > 12 THEN PRINT 00$: GOTO 750
65 760 DI = VAL ( MID$ (MD$,2 * M - 1,2)):DI = DI +
    (M = 2) * LY:DI$ = STR$ (DI):DI$ = RIGHT$ (DI
    $,2)

```

```

3B 770 PRINT : PRINT "DAY (1-"DI$") " ; GOSUB 2240:
    IF I$ < > "" THEN D = VAL (I$)
8A 775 IF D < 1 OR D > DI THEN PRINT 00$: GOTO 770
F2 780 H$ = MID$ (A$, (M * 3) - 2, 3) + " ": PRINT : P
    RINT "LATITUDE (0-90)"; GOSUB 2240: IF I$ <
    > "" THEN LL = VAL (I$)
F8 784 GOSUB 4500
E9 790 IF ABS (LL) > 90 THEN PRINT 00$: GOTO 780
E7 800 PRINT : HTAB 5: GOSUB 1670: GOSUB 1295: GOSUB
    1550: PRINT : PRINT : GOSUB 2020: IF I$ = "N
    " THEN 720
8D 820 D2 = VAL ( MID$ (M$, (M * 3) - 2, 3)) + D: GOSU
    B 1640: IF M > 2 THEN D1 = D1 + LY: Y1 = Y1 +
    LY
2D 830 D3 = D2 - 185: IF M = 3 AND D < 20 THEN D2 =
    D2 + LY: D3 = D3 + LY
F0 840 S = 0: IF D3 < = 0 THEN A = 180 * D2 / 185: G
    OTO 860
E2 850 A = 180 * D3 / (180 + 2Y) + 180
82 860 IF A < > 180 THEN S = 23.43333333 * ( SIN (D9
    * D2 * 180 / 185))
04 870 IF A > 180 THEN S = - 23.43333333 * ( SIN (D9
    * D3))
E9 880 IF A > = 360 THEN A = A - 360
83 885 A = FN R(A)
E1 890 S = FN R(S): A1 = ( SGN (LL) + (LL = 0)) * S +
    90 - ABS (LL): A1 = FN R(A1): GOSUB 1250: GOS
    UB 1200
87 895 W = 2 - (LL < 0): IF A1 > 90 THEN A1 = 180 -
    A1: W = 3 - W
25 900 HOME : VTAB 2: GOSUB 1550: PRINT : PRINT "----
    -----"
7E 910 PRINT : PRINT "DAY OF THE YEAR-----
    "; D1
99 920 PRINT "SUN'S GEOCENTRIC ANGLE-----      "; A; "0"
8E 930 PRINT "SUN'S DECLINATION-----          "; S; "0"
E3 940 PRINT "SUN'S ALTITUDE AT NOON-----      "; A1; "0
    "; D$(W)
FF 950 PRINT "SUN'S RIGHT ASCENSION -----      "; A3$
E2 960 PRINT "R.A. AT 9:00PM-----              "; A5$
32 970 PRINT "MOON'S AGE-----                  "; M1; "D
    Y"
08 980 PRINT "MOON'S ELONGATION-----            "; M8; "0
    "; L$
2D 990 PRINT "MOON'S PHASE - "PH$(M3)
63 1000 VTAB 17: PRINT "-P- PLANET TABLE , -D- NEW DA
    TE": GOTO 1700
F3 1010 HOME : HTAB 11: PRINT "** PLANET TABLE **":
    VTAB 3: GOSUB 1550: S1 = 1

```



```

CE 1020 VTAB 5: HTAB 1: PRINT "PLANET   DIST.   ANG.
      W/ SUN   R.A"
AA 1030 VTAB 6: PRINT "-----"
      "-----"
D4 1040 FOR X = 1 TO 6:A2 = Y1 / P(X,2) - INT (Y1 /
      P(X,2)):Q3 = 1
19 1050 A2 = (A2 * 360) + P(X,1): IF A2 > 360 THEN A
      2 = A2 - 360
01 1060 E = 180 + A: IF E > 360 THEN E = E - 360
FD 1070 E1 = ABS (E - A2): IF E1 > 180 THEN E1 = 360
      - E1
24 1080 GOSUB 1310:E1 = E1 * D9:P5 = P(X,3): IF X =
      3 THEN GOSUB 1750
10 1090 P(X,4) = SQR (1 + P5 ^ 2 - 2 * 1 * P5 * COS
      (E1)):XX = ((P5 ^ 2 - 1 - P(X,4) ^ 2) / (-
      2 * P(X,4)))
70 1100 P(X,5) = - ATN (XX / SQR (- XX * XX + 1)) +
      ATN (1) * 2:P(X,4) = INT (P(X,4) * 93 + .5)
      :P(X,5) = P(X,5) / D9
56 1110 P(X,5) = FN S(P(X,5)):Q1$ = STR$ (P(X,4)):Q2
      $ = STR$ (P(X,5))
08 1120 Q1 = LEN (Q1$):Q2 = LEN (Q2$): GOSUB 1410
08 1130 PRINT P$(X); TAB( 14 - Q1);Q1$; TAB( 24 - Q2
      );Q2$;: IF Q3 = - 1 THEN PRINT "0W";
0A 1140 IF Q3 = 1 THEN PRINT "0E";
78 1150 GOSUB 1460:Q4$ = STR$ (Q4):Q5$ = STR$ (Q5):
      IF Q5 < 10 THEN Q5$ = "0" + RIGHT$ (Q5$,1)
90 1160 Q5$ = RIGHT$ (Q5$,2):Q4$ = Q4$ + ":" + Q5$:Z
      = LEN (Q4$)
71 1170 PRINT TAB( 28)Q4$ TAB( 36 - Z)Q4$: NEXT : VT
      AB 14: PRINT "*" - VISIBLE AT 9 P.M."
82 1180 VTAB 17: PRINT "SUN'S R.A.-----" SPC( Q8)
      A3$: PRINT "R.A. AT 9:00PM ----" SPC( Q9)A5$
15 1190 VTAB 21: PRINT "-S- FOR DAYS SKY -D- FOR NEW
      DATE": GOTO 1700
04 1200 A2 = K1 * A / 360: IF A2 > K1 THEN A2 = A2 -
      K1
77 1210 A3 = INT (A2 / 60):A4 = A2 - A3 * 60:A5 = A3
      + 9: IF A5 > 23 THEN A5 = A5 - 24
27 1220 A4 = INT (A2 - A3 * 60 + .5): IF A4 = 60 THE
      N A4 = 0:A3 = A3 + 1
93 1230 IF A3 = 24 THEN A3 = 0
0C 1240 AA = A3 * 60 + A4: GOTO 1560
8D 1250 M1 = ((Y1 / M9) - INT (Y1 / M9)) * M9 + 10:
      IF M1 > M9 THEN M1 = M1 - M9
BC 1260 GOSUB 2050:M8 = 360 * M2: IF M8 > 180 THEN L
      $ = "W"
B4 1270 IF M8 < = 180 THEN L$ = "E"
EE 1280 IF M8 > 180 THEN M8 = 360 - M8
56 1290 M1 = FN R(M1):M8 = FN R(M8)

```

```

00 1295 YY = INT (7 * (Y1 / 7 - INT (Y1 / 7)) + .2):
    IF YY = 0 THEN YY = 7
40 1300 K$ = MID$ (J$, (YY * 3) - 2, 3): RETURN
14 1310 Q3 = 0: Q1 = E + 180: IF Q1 > 360 THEN 1350
61 1320 IF A2 > E AND A2 < Q1 THEN 1340
02 1330 Q3 = 1: RETURN
10 1340 Q3 = - 1: RETURN
6E 1350 Q1 = Q1 - 360: IF A2 < = 360 AND A2 > E THEN
    1340
80 1360 IF Q3 < > 0 THEN RETURN
6A 1370 IF A2 > 0 AND A2 < = Q1 THEN 1340
95 1380 IF Q3 < > 0 THEN RETURN
44 1390 IF A2 > Q1 THEN 1330
09 1400 RETURN
F8 1410 Q5 = Q3 * P(X, 5) * 4 + AA: IF Q5 < 0 THEN Q5
    = Q5 + K1
BC 1420 IF Q5 > K1 THEN Q5 = Q5 - K1
92 1430 P(X, 6) = Q5: Q4 = INT (Q5 / 60): Q5 = INT (Q5
    - Q4 * 60 + .5): IF Q5 = 60 THEN Q5 = 0: Q4 =
    Q4 + 1
29 1440 IF Q4 = 24 THEN Q4 = 0
ED 1450 RETURN
08 1460 SU = A5 * 60 + A4: PS = SU + 360: MS = SU - 36
    0: IF PS > K1 THEN PS = PS - K1
96 1470 IF MS < 0 THEN MS = MS + K1
5A 1480 IF MS > PS THEN 1510
26 1490 IF P(X, 6) < PS AND P(X, 6) > MS THEN 1540
02 1500 QQ$ = " ": RETURN
DE 1510 IF P(X, 6) < K1 AND P(X, 6) > MS THEN 1540
46 1520 IF P(X, 6) < PS THEN 1540
6A 1530 GOTO 1500
A3 1540 QQ$ = "*": RETURN
80 1550 PRINT K$"— "H$;D", "Y;" "": IF LL < 10 THEN
    PRINT " ";
AB 1555 PRINT ABS (LL); LL$;: RETURN
EF 1560 A3$ = STR$ (A3): A3$ = RIGHT$ (A3$, 2): A4$ = S
    TR$ (A4): A4$ = RIGHT$ (A4$, 2)
42 1570 IF A4 < 10 THEN A4$ = "0" + RIGHT$ (A4$, 1)
30 1580 A3$ = A3$ + ":" + RIGHT$ (A4$, 2): A5$ = STR$
    (A5): A5$ = RIGHT$ (A5$, 2) + ":" + A4$
EB 1590 Q8 = 7 - LEN (A3$): Q9 = 7 - LEN (A5$): RETUR
    N
5D 1600 LY = 0: IF Y / 4 = INT (Y / 4) THEN LY = 1
49 1610 IF Y / 100 = INT (Y / 100) AND Y / 400 < > I
    NT (Y / 400) THEN LY = 0
CF 1620 IF Y / 1000 = INT (Y / 1000) AND Y / 4000 =
    INT (Y / 4000) THEN LY = 0
E9 1630 RETURN
4B 1640 Y9 = Y + 1: IF Y9 / 4 = INT (Y9 / 4) THEN ZY
    = 1

```

```

08 1650 IF Y9 / 100 = INT (Y9 / 100) AND Y9 / 400 <
    > INT (Y9 / 400) THEN ZY = 0
08 1660 IF Y9 / 1000 = INT (Y9 / 1000) AND Y9 / 4000
    = INT (Y9 / 4000) THEN ZY = 0
08 1670 Y1 = Y - 1977:Y1 = Y1 * 365 + INT (Y1 / 4) +
    D1: IF Y < 2000 THEN 1690
0C 1680 Y1 = Y1 - INT ((Y - 2001) / 100) + INT ((Y -
    2001) / 400) - INT ((Y - 1) / 4000)
02 1690 RETURN
5D 1700 GET I$
F1 1710 IF I$ = "D" THEN 720
F9 1720 IF (I$ = "S" OR I$ = "T") AND S1 = 1 THEN 23
    0
A9 1730 IF I$ = "P" THEN 1010
06 1735 IF I$ = "L" AND S1 = 1 THEN 4550
76 1740 GOTO 1700
EB 1750 P5 = 1.376344086:K5 = A2 * 4
97 1760 K5 = ABS (K5 - 1233.73) * 90 / K1:K5 = K5 *
    D9:K5 = SIN (K5) * .322581224:P5 = P5 + K5:
    RETURN
B9 1770 IF CC < = 1 THEN CC = CC + 84
47 1780 CD$ = MID$ (CC$,CC - 1)
0F 1785 IF MID$ (CD$,2,1) < > " " AND MID$ (CD$,3,1)
    = " " THEN CD$ = " " + CD$
3D 1786 IF MID$ (CD$,41,1) = " " AND MID$ (CD$,42,1)
    < > " " THEN CD$ = MID$ (CD$,2)
03 1788 CD$ = MID$ (CD$,2,40): RETURN
8D 1790 DATA 365.26,29.53059,59.818184,42.719626,262
    .3644,52.916763
91 1800 DATA 134.69697,218.79464,87.97,224.7,686.98
39 1810 DATA 4332.79813,10759.7195,30686.5884
25 1820 DATA "MERCURY",.3871,"VENUS",.7233,"MARS",1.
    5237,"JUPITER",5.2028
06 1830 DATA "SATURN",9.5308,"URANUS",19.182
A5 1890 DATA "SA","SC","LI","VI","LE","CA","GE","TA"
    ,"AR","PI","AQ","CP"
15 1900 DATA "NEW","WAXING CRESCENT","1ST QUARTER","
    WAXING GIBBOUS","FULL"
E4 1910 DATA "WANING GIBBOUS","3RD QUARTER","WANING
    CRESCENT"
05 1920 DATA 1770,1719,1620,1500,1418,1365,1335,1310
    ,1290,1275,1260
4A 1930 DATA 1238,1220,1200,1178,1115,915,720,660,64
    0,625,610
D4 1940 PRINT CHR$ (17): HOME : VTAB 7: HTAB 12: PRI
    NT "***** SKYSCAPE *****"
F7 1950 RETURN
AC 2020 PRINT "-N- TO RE-INPUT OR RETURN TO CONTINUE
    "

```



```

6F 2030 GET I$: RETURN
36 2050 M2 = M1 / M9: IF M1 < 1 OR M1 > 28.5 THEN M3
    = 1
CA 2060 IF M1 >= 1 AND M1 < 6.9 THEN M3 = 2
36 2070 IF M1 <= 8.0 AND M1 >= 6.9 THEN M3 = 3
D9 2080 IF M1 > 8.0 AND M1 < 14.2 THEN M3 = 4
BB 2090 IF M1 >= 14.2 AND M1 <= 15.2 THEN M3 = 5
69 2100 IF M1 > 15.2 AND M1 < 21.6 THEN M3 = 6
6F 2110 IF M1 >= 21.6 AND M1 <= 22.6 THEN M3 = 7
34 2120 IF M1 > 22.6 AND M1 <= 28.5 THEN M3 = 8
E8 2130 RETURN
12 2140 B$ = "": IF Y < > 1985 AND Y < > 1986 THEN 5
    60
41 2150 IF (Y = 1985 AND D1 < 305) OR (Y = 1986 AND
    D1 > 149) THEN 560
AB 2160 HD = D1 + 365: IF HD > 516 THEN HD = HD - 36
    5
D2 2170 H1 = (HD - 295) / 10: HD = INT (H1): H1 = H1 -
    HD
FA 2180 T4 = HC(HD) - HC(HD + 1): T4 = HC(HD) - H1 *
    T4: IF T4 > 1440 THEN T4 = T4 - 1440
AB 2190 GOSUB 610: IF Y9 = 999 THEN 560
3A 2200 GOSUB 700: IF T4 > 1115 AND T4 < 1200 THEN U
    9 = U9 + 1
07 2210 IF T4 > 1290 THEN U9 = U9 - 1
A3 2220 IF T4 > 615 AND T4 < 1115 THEN U9 = U9 + 2
6E 2230 U(7) = U9: B$ = CHR$ (PP(7)) + "HALLEY'S COME
    T": GOTO 530
4D 2240 INPUT I$: RETURN
BB 2250 VTAB 17: PRINT CD$;: RETURN
2F 4000 GOSUB 700
27 4005 IF LL >= 0 THEN U9 = LC + 9 + U9: GOTO 4008
EE 4006 U9 = LC + 9 - U9: Y9 = 39 - Y9
15 4008 RETURN
AB 4500 LL$ = "0N": IF LL < 0 THEN LL$ = "0S"
F3 4510 L1 = ABS (LL): IF L1 < 24 THEN L1 = 40
44 4515 LC = INT ((L1 - 40) / 7 + .5): D1 = VAL ( MID
    $ (D$, (M * 3) - 2, 3)) + D
EA 4530 RETURN
1B 4550 HOME : VTAB 2: HTAB 7: PRINT "***** SKY
    SCAPE *****": VTAB 4: PRINT "LATITUDE C
    HANGE"
E9 4555 PRINT "-----"
AC 4560 VTAB 8: PRINT "ENTER NEW LATITUDE";: GOSUB 2
    240: IF I$ < > "" THEN LL = VAL (I$)
C9 4565 IF ABS (LL) > 90 THEN PRINT 00$: GOTO 4560
E2 4570 GOSUB 2020: IF I$ = "N" THEN 4550
AB 4580 GOSUB 4500: I$ = "S": GOTO 1720
28 5000 CI = 1: C2$ = ""

```

```
25 5010 C1$ = MID$ (CD$,CI,1): IF C1$ < > " " THEN 5030
AA 5020 C2$ = C1$ + C2$:CI = CI + 1: GOTO 5040
85 5030 C1$ = MID$ (CD$,CI,2):C2$ = C1$ + C2$:CI = C
    I + 2
F8 5040 IF CI < 41 THEN 5010
58 5050 CD$ = C2$: RETURN
```

Puzzler

Mark Tuttle and Kevin Mykytyn

Apple version by Kevin Martin

Test your skill in pattern matching and visualization with "Puzzler," a game of utmost concentration. For all Apple II series computers using either DOS 3.3 or ProDOS.

How good are you at recognizing patterns? Many intelligence tests measure this important conceptual skill. "Puzzler" challenges your ability to find matching patterns in a background of similar shapes. It displays two puzzle grids composed of multicolored blocks, with both grids containing exactly the same blocks. Those in the left grid, though, have been scrambled. Your job is to rearrange the blocks in the left puzzle grid until they match those on the right. You must solve the puzzle before time runs out.

Type in the program, and don't forget to save a copy of the game before you run it.

Puzzle Building

Puzzler begins by letting you choose the size of the puzzle grid. Enter values for the number of rows and columns in the grid. The maximum puzzle size is seven rows and seven columns. Of course, larger puzzles are more difficult to solve than small ones. Next, enter the number of colors the puzzle will use. Two-color puzzles are the easiest. You can use up to 15 colors. Again, the more you choose, the harder your job becomes.

Puzzler then spends a short time building the two grids. Since the blocks are arranged at random, each new puzzle is different from the last. While you try to solve the puzzle, the computer keeps track of the time and alerts you when the puzzle is solved or when time runs out. The time limit depends on the size of the puzzle.

Puzzler allows three different operations. You can move within the puzzle grid from one block to another, pick up a block and move it to a new position, or rotate a block in its current position. Use the I, J, K, and L keys to move up, left, down, and right, respectively, in the grid. Your position is indicated by small white highlights in the corners of a block. To

pick up a block, press the space bar once. The cursor or arrow changes color to show that you're carrying the piece. Then move to the position where you want to place the block, and press the space bar once again. The block in the current position trades positions with the block you're carrying.

Each block consists of four colored squares. To rotate a block in its current position, press the space bar twice. The block rotates 90 degrees. You may rotate a block as many times as you want.

Continue moving and rotating blocks until both puzzle grids match. Every block must match in color and be turned in the right direction.

Timing

The allotted time to solve a puzzle depends on its complexity. The rapidly decreasing time value you see on the screen is calculated by doubling the number of rows and columns, multiplying these numbers together, then multiplying that by 75. If you're trying to solve a puzzle three rows wide by three columns high, then, you'll see 2700 on the screen. Colors don't affect the allowed time.

If you find that there's not enough time to complete a puzzle (even the simplest), change line 810 in the program so that $T = NR * NC * \text{value higher than } 75$. The higher the number, the more time you'll have with any level puzzle.

Puzzler

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```

29 100 A$ = "": IF PEEK (24576) = 162 THEN 140
55 110 FOR I = 24576 TO 24872
82 120 READ A: POKE I,A
FE 130 NEXT
C5 140 HIMEM: 23576
50 150 GOSUB 550
50 160 IF T = 0 THEN VTAB 21: PRINT TAB( 14)"OUT OF
    TIME": GOTO 380
DB 170 HTAB 17: VTAB 23: PRINT T;" "
47 180 T = T - 1
2A 190 IF PEEK ( - 16384) < 128 THEN 160
90 200 GET C$: IF (C$ < "I" OR C$ > "L") AND C$ < >
    " " THEN 160
81 210 R = R - (C$ = "I") + (C$ = "K")
59 220 C = C - (C$ = "J") + (C$ = "L")
7E 230 IF R < 0 THEN R = 0

```

```

9E 240 IF R >= R3 THEN R = R3 - 1
AF 250 IF C < 0 THEN C = 0
DE 260 IF C >= C3 THEN C = C3 - 1
61 270 POKE 773,X1 + C * 2 - 1: POKE 772,Y1 + R * 2
    - 1: CALL 24671
C0 280 IF C$ < > " " THEN 160
46 290 IF F = 0 THEN 440
0A 300 F = 0: IF RR = R AND CC = C THEN GOSUB 510: G
    OTO 320
4B 310 GOSUB 460
02 320 CALL 24691
C4 330 POKE 768,X1: POKE 769,Y1: CALL 24576
B0 340 POKE 773,X1 + C * 2 - 1: POKE 772,Y1 + R * 2
    - 1: POKE 774,255: CALL 24753
C3 350 IF A$ < > B$ THEN 160
DA 360 CALL 24691
74 370 HOME : PRINT TAB( 16);"CORRECT!"
59 380 HTAB 13: VTAB 22: PRINT "PRESS ANY KEY."
1C 390 HTAB 17: VTAB 23: PRINT T
0F 400 POKE - 16368,0
0F 410 IF PEEK ( - 16384) < 128 THEN 410
D4 420 GET A$
AC 430 RUN
3D 440 F = 1:RR = R:CC = C: POKE 773,X1 + C * 2 - 1:
    POKE 772,Y1 + R * 2 - 1: POKE 774,119: CALL
    24671
9D 450 GOTO 160
8B 460 AA = SS + 2 * NC * RR + 2 * CC:A = SS + 2 * N
    C * R + C * 2
04 470 D = PEEK (A): POKE A, PEEK (AA): POKE AA,D
C0 480 D = PEEK (A + 1): POKE A + 1, PEEK (AA + 1):
    POKE AA + 1,D
9B 490 D = PEEK (A + NC): POKE A + NC, PEEK (AA + NC
    ): POKE AA + NC,D
A4 500 D = PEEK (A + NC + 1): POKE A + NC + 1, PEEK
    (AA + NC + 1): POKE AA + NC + 1,D: RETURN
4E 510 A = SS + 2 * NC * R + C * 2
4B 520 D = PEEK (A): POKE A, PEEK (A + NC)
4E 530 POKE A + NC, PEEK (A + NC + 1)
6F 540 POKE A + NC + 1, PEEK (A + 1): POKE A + 1,D:
    RETURN
62 550 TEXT : HOME
0B 560 PRINT TAB( 16);"PUZZLER"
F2 570 INPUT "NUMBER OF ROWS (2-7):";R3
B8 580 IF R3 < 2 OR R3 > 7 THEN 570
77 590 INPUT "NUMBER OF COLUMNS (2-7):";C3
6F 600 IF C3 < 2 OR C3 > 7 THEN 590
BC 610 INPUT "NUMBER OF COLORS (2-15):";CO
FD 620 IF CO < 2 OR CO > 15 THEN 610
0D 630 PRINT "PLEASE WAIT..."

```

```

FC 640 NR = 2 * R3:NC = 2 * C3
7E 650 FOR A = 1 TO NR * NC:B = INT ( RND (1) * CO +
    1):A$ = A$ + CHR$ (B + B * 16): NEXT :B$ = A
    $
A2 660 A = PEEK (105) + PEEK (106) * 256
CF 670 SS = PEEK (A + 3) + PEEK (A + 4) * 256
54 680 X1 = 10 - C3:Y1 = 9 - R3:X2 = X1 + 20
50 690 POKE 24600, PEEK (A + 3): POKE 24601, PEEK (A
    + 4)
% 700 POKE 768,X2: POKE 769,Y1: POKE 770,NC: POKE 7
    71,NR + Y1
49 710 GR
F4 720 CALL 24576
ED 730 FOR R = 0 TO R3 - 1: FOR C = 0 TO C3 - 1:B =
    INT ( RND (1) * 4)
4B 740 IF B THEN GOSUB 510:B = B - 1: GOTO 740
CA 750 NEXT : NEXT
42 760 FOR R = 0 TO R3 - 1: FOR C = 0 TO C3 - 1
B4 770 RR = INT ( RND (1) * R3):CC = INT ( RND (1) *
    C3): GOSUB 460: NEXT : NEXT
D2 780 POKE 768,X1: POKE 769,Y1: CALL 24576
5E 790 HOME : PRINT TAB( 16);"PUZZLER"
20 800 POKE 772,Y1 - 1: POKE 773,X1 - 1: POKE 774,25
    5: CALL 24753
B3 810 R = 0:C = 0:T = NR * NC * 75: RETURN
0E 820 DATA 162,0,172,1,3,185
C9 830 DATA 47,96,24,109,0,3
93 840 DATA 133,251,185,71,96,105
72 850 DATA 0,133,252,160,0,189
BB 860 DATA 140,89,145,251,232,200
AF 870 DATA 204,2,3,208,244,238
7B 880 DATA 1,3,173,1,3,205
DD 890 DATA 3,3,208,212,96,0
7E 900 DATA 128,0,128,0,128,0
39 910 DATA 128,40,168,40,168,40
F5 920 DATA 168,40,168,80,208,80
05 930 DATA 208,80,208,80,208,4
4A 940 DATA 4,5,5,6,6,7
21 950 DATA 7,4,4,5,5,6
1E 960 DATA 6,7,7,4,4,5
3E 970 DATA 5,6,6,7,7,32
CA 980 DATA 115,96,76,177,96,24
E3 990 DATA 121,47,96,133,251,185
4C 1000 DATA 71,96,105,0,133,252
08 1010 DATA 96,172,7,3,173,8
FA 1020 DATA 3,32,101,96,160,0
67 1030 DATA 162,0,189,9,3,145
D1 1040 DATA 251,232,200,200,200,189
A5 1050 DATA 9,3,145,251,232,173
BC 1060 DATA 7,3,24,105,3,141

```


31 1070 DATA 7,3,168,173,8,3
72 1080 DATA 32,101,96,160,0,189
9C 1090 DATA 9,3,145,251,232,200
CD 1100 DATA 200,200,189,9,3,145
BD 1110 DATA 251,232,96,172,4,3
CA 1120 DATA 140,7,3,173,5,3
09 1130 DATA 141,8,3,32,101,96
27 1140 DATA 160,0,162,0,177,251
67 1150 DATA 157,9,3,232,41,15
BD 1160 DATA 145,251,173,6,3,41
51 1170 DATA 240,17,251,145,251,200
1F 1180 DATA 200,200,177,251,157,9
FE 1190 DATA 3,232,41,15,145,251
11 1200 DATA 173,6,3,41,240,17
3A 1210 DATA 251,145,251,173,4,3
FB 1220 DATA 24,105,3,141,4,3
AF 1230 DATA 168,173,5,3,32,101
2F 1240 DATA 96,160,0,177,251,157
DF 1250 DATA 9,3,232,41,240,145
31 1260 DATA 251,173,6,3,41,15
0F 1270 DATA 17,251,145,251,200,200
EE 1280 DATA 200,177,251,157,9,3
96 1290 DATA 232,41,240,145,251,173
C0 1300 DATA 6,3,41,15,17,251
23 1310 DATA 145,251,96

3

Applications



Apple SpeedCalc

Kevin Martin

This professional-quality spreadsheet program for all Apple II series computers with either DOS 3.3 or ProDOS is written completely in high-speed machine language. Apple SpeedCalc has all the important features you'd expect from a commercial spreadsheet program. In addition, its data files can be merged into text files created with the Apple SpeedScript word processor published in COMPUTE! magazine in 1985. Apple SpeedCalc requires a disk drive; a printer is optional but recommended.

Have you ever planned a budget for your home or office? If so, you probably used some sort of worksheet divided into rows and columns. Perhaps you wrote the months of the year along the top of the sheet and listed categories for earnings and expenses along one side. After entering data for each category and month of the year, you could calculate total income figures by adding or subtracting numbers in each of the sheet's "cells."

That's a classic example of a worksheet. It lets you enter and organize data, then perform calculations that produce new information. A *spreadsheet* program is an electronic version of the familiar paper worksheet. Since it does all the calculations for you at lightning speed, an electronic spreadsheet is far more convenient than its paper counterpart. And spreadsheet programs also offer editing features that let you enter and manipulate large amounts of data with a minimum of effort.

Apple *SpeedCalc* is an all machine language spreadsheet program for Apple II computers with either DOS 3.3 or ProDOS. Though relatively compact in size, *SpeedCalc* is fast, easy to use, and has many of the features found in commercial spreadsheet programs.

Even better, the "*SpeedScript* File Converter" program published along with *SpeedScript* lets you merge your *SpeedCalc* files into word processing documents created with *SpeedScript*, COMPUTE!'s popular word processor (see *COMPUTE!* magazine, July 1985, or *SpeedScript: The Word Processor for Apple Personal Computers*, published by COMPUTE! Books).

Working together, *SpeedCalc* and *SpeedScript* make a powerful team. You can merge a chart of sales figures into a com-

pany report, create a table of scientific data for a term paper, and manipulate numeric information in many other ways. In a sense, a spreadsheet program brings to arithmetic all the flexibility and power that a word processor brings to writing.

Preparing the Program

Although Apple *SpeedCalc* is small in comparison to similar commercial programs, it is one of the longest programs COMPUTE! Books has ever published. Fortunately, the "Apple MLX" machine language entry utility makes it easier to type a program of this size. Be sure to read the MLX article in Appendix C carefully and have the program saved on disk before you begin.

There are two separate versions of Apple *SpeedCalc*: Program 1 is for Apple computers with DOS 3.3, and Program 2 is for Apples with ProDOS. Be sure to type the correct version for your system, since the DOS 3.3 version doesn't work with ProDOS and vice versa.

Since the DOS 3.3 version of *SpeedCalc* resides in the same area of memory normally used by BASIC programs, you must relocate the BASIC program storage area *before* loading MLX to enter the data for *SpeedCalc*. If you're using DOS 3.3, enter the line below in direct mode (without a line number) and press Return:

```
POKE 104,38:POKE 9728,0:NEW
```

Then load and run MLX.

If you're using ProDOS, no special actions are required before loading and running MLX.

When you first run MLX, you are asked for a starting address and an ending address for the data you will be entering. Here are the addresses you need to enter *SpeedCalc* with MLX:

DOS 3.3:

Starting address: 07FA

Ending address: 24F9

ProDOS:

Starting address: 2000

Ending address: 3D67

After you finish typing, be sure to save at least one copy before attempting to run *SpeedCalc* for the first time. To start the DOS 3.3 version, first enter BLOAD SPEEDCALC (replace SPEEDCALC with the appropriate filename if you used some

other name when saving the program). After the program loads, simply type RUN as you would for a BASIC program.

To start the ProDOS version of *SpeedCalc*, first boot ProDOS, then enter -SPEEDCALC (replace SPEEDCALC with the appropriate filename if you used some other name when saving the program). This removes the BASIC interpreter and lets *SpeedCalc* take over the system.

If you're using an Apple IIe or IIc, be sure the Caps Lock key is down: *SpeedCalc* doesn't accept lowercase text input.

The Apple *SpeedCalc* Screen

SpeedCalc uses the top line of the screen as the *command line*. This is where *SpeedCalc* displays messages and asks you questions.

Screen lines 2-4 are the *input buffer* area. This is the work area where you enter and edit data. As you'll see in a moment, the input buffer also displays the data contained in the current cell. The work area cursor is a reverse less-than sign (<). When the cursor is solid (nonblinking), *SpeedCalc* is waiting for a command or for data to be entered. After a character of data has been entered, the cursor begins blinking. While the cursor is blinking, most *SpeedCalc* commands (except for the cursor movement keys) are deactivated until you press Return to enter the data into the worksheet.

The lower 20 screen lines are your window into the spreadsheet. Though the spreadsheet contains many rows and columns, only a few can fit on the screen at one time. By scrolling the screen back and forth with the cursor, you can move the display window to any part of the spreadsheet.

The *SpeedCalc* worksheet consists of 50 vertical columns labeled with letters (AA, AB, ..., BX) and 200 horizontal rows numbered 1-200. The rectangle where a row and column intersect is called a *cell*. Cells are where you store data. With 50 columns and 200 rows, the *SpeedCalc* spreadsheet has a maximum of 10,000 (50*200) cells. Due to memory limitations, however, only about a third of these can actually contain data. But you may spread out the data over all 10,000 cells if necessary, depending on the format you need.

Moving the Cursor

Each cell is identified with the letters of its column and the number of its row. For example, the cell at the extreme upper-

left corner of the sheet is called AA1, since it's in column AA and row 1. The cell below that is AA2. Moving one cell to the right from AA2 puts you in cell AB2, and so on.

Your current position in the spreadsheet is shown by the highlighted cursor. The simplest way to move around the sheet is with the cursor keys (on the Apple II or II+, use Ctrl-K to move up and Ctrl-J to move down). Another way to move the cursor is with Ctrl-@ (Ctrl-Shift-P for the Apple II or II+, or Ctrl-2 for the Apple IIe and IIc.) Press Ctrl-@ once to "home" the cursor on the current screen: The cursor moves to the upper-left cell. Quickly press Ctrl-@ twice to move the cursor to cell AA1, the home position for the entire sheet.

SpeedCalc also has a *goto* command for moving the cursor over long distances. When you press Ctrl-G, the command line displays GOTO:, followed by an underline cursor. The underline cursor generally indicates that *SpeedCalc* is waiting for data—in this case, it expects the name of the cell where you wish to go. If you type BA188 at this point, *SpeedCalc* moves the cursor to the cell at column BA, row 188, adjusting the screen window as needed. Take a few moments to practice moving around the spreadsheet with all three methods; you'll be using them a lot. In a later section, we'll discuss how to change the size and format of a cell.

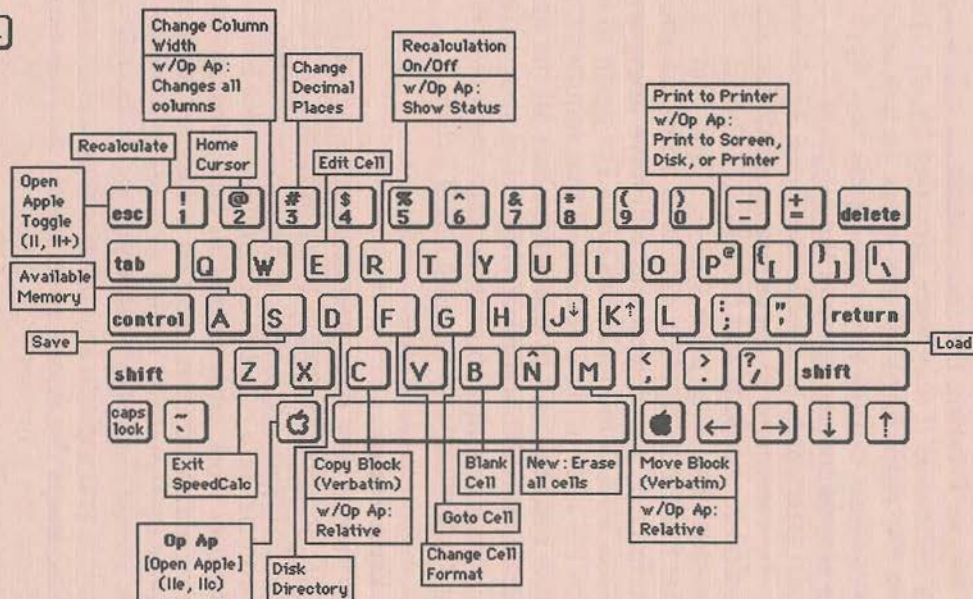
Keyboard Commands

SpeedCalc offers many different commands, a few of which are entered by pressing one key. However, most commands are entered by pressing Ctrl (Control) along with another key. Ctrl-G, as you've seen, is the *goto* command. Ctrl-A displays the amount of free memory available, Ctrl-B blanks a cell, and so on. The most drastic command is Ctrl-X, which exits *SpeedCalc* and reboots the system. Since this effectively erases all data in memory, *SpeedCalc* prompts you with ARE YOU SURE Y/N? before it shuts down. To cancel the command, simply type N (or any key other than Y).

A few commands require you to press three keys at once. This sounds more awkward than it really is, since two of the three keys are Open Apple and Ctrl (the Open Apple key is the one just to the left of the space bar on the Apple IIe and IIc). For instance, the *relative copy* command is performed by pressing Open Apple-Ctrl-C (hold down both the Open Apple and Ctrl keys, then press C).

SpeedCalc Keyboard Reference

Use **control** or **CTRL**
with most commands
Apple IIc Keyboard Shown.
Apple IIe, II+ similar



The older Apple II and II+ models don't have an Open Apple key, so the Esc key is programmed to act as an Open Apple toggle. Pressing Esc once makes all following keypresses behave as if they were preceded by an Open Apple. Pressing Esc again turns off this effect. In this article, wherever the instructions call for the Open Apple key, Apple II and II+ owners should instead precede the keypress with Esc, then use Esc again afterwards to disable the Open Apple toggle. For example, the command to check the recalculation status is Open Apple-Ctrl-R. Apple II and II+ owners should instead press (and release) Esc, then press Ctrl-R. There's no visible indication that the Open Apple toggle is in effect, so you must use Esc carefully or your keypresses will have unexpected results. For safety, always remember to press Esc again to toggle this function off after using a command that requires Open Apple. The table near the end of this article lists all the *SpeedCalc* commands, and the accompanying figure shows the keyboard layout with a description of what each key does. We'll be discussing each command in more detail below.

Three Data Types

Before entering any data, you must know what kind of data *SpeedCalc* accepts. There are three different types: *numbers*, *text*, and *formulas*. Let's look at each type in turn.

Numeric data consists of numbers—the basic stuff that spreadsheets work with. *SpeedCalc* has a few simple rules for numeric data: A number must be a decimal value (base 10, not hexadecimal) composed of one or more digits from 0 through 9, with an optional plus or minus sign. A decimal point is also optional. If you include any other characters in numeric input, *SpeedCalc* treats the entire input as *text data* (as explained below). Thus, the numbers 123, 0.001, and -65535 are valid numeric data. The number 65,535 is invalid because it includes a comma.

The allowable range for numbers in Apple *SpeedCalc* is the same as for Applesoft, roughly $-1.7\text{E}38$ to $+1.7\text{E}38$. If a calculation produces a number outside the allowable range, you'll see the message *ERROR* in the cell containing the formula. This doesn't happen very often, since *SpeedCalc* won't let you enter a number more than 36 digits long, and there's rarely a need to use such large numbers unless you're tracking the national debt.

Although an input value can be up to 36 digits long, numbers in *SpeedCalc* calculations are accurate only to nine digits. This must be taken into account when you're doing any calculation involving large values. For example, you can enter the value 1122334455.66 into a cell, and the cell holds the value with no rounding. However, if you use the value from that cell in a formula, the value is rounded to nine significant digits—1122334460.00—and the result of the calculation is accurate only for the first nine digits.

You can enter values in scientific notation by following a number with the letter *E* and the appropriate power of 10. For example, you can enter 1,234,000 as 1.234E06. However, *SpeedCalc* never uses scientific notation itself, no matter how big the number you enter. Scientific notation should generally be avoided since values outside the Apple's maximum range will crash the program. (Press Ctrl-Reset to recover.)

For example, let's enter the number 123 in cell AA1. No special commands are required to enter data: Just move the cursor to AA1 and begin typing. The blinking reverse < sign shows the end of the data. While you're entering the number, it appears only in the input buffer near the top of the screen (the blinking underline shows your cursor position). As soon as you press Return, the number 123.00 appears in AA1 and the letter *N* appears at the upper right of the screen. The *N* signifies *numeric*, meaning that *SpeedCalc* has accepted the entry as valid numeric data. Move the cursor to a vacant cell, then move it back to AA1. The input buffer displays whatever data is found in the cell under the cursor. When the current cell is empty, the buffer is empty as well.

If you want to change anything during data entry, press the Esc key. Esc always deletes the character before the cursor (or has no effect if the cell is empty). Later on, we'll explain how to edit existing data. Use Esc carefully; remember that when you're not editing (when the cursor is not blinking) Esc acts as an Open Apple toggle. On the Apple IIe and IIc, you can (and should) use the Delete key instead of Esc.

As you've seen, pressing Return enters a data item into the current cell. You can also end the input by pressing a cursor key. The data is entered as if you had pressed Return, and the cursor moves in the indicated direction. This feature is handy for entering a lot of data: Simply type the entry, move the cursor to the next cell, enter more data, and so on.

Text data is not data in the strict sense, since *SpeedCalc* doesn't use it in calculations as it does numbers and formulas. Text data is there only to help people understand what the other data means. Text may consist of comments, titles, column headings, subheadings, or whatever you need to interpret the numbers and formulas. As an example, move the cursor to cell AA2 (just under AA1) and type the following line:

THIS IS A PIECE OF TEXT DATA.

You can use the Esc key (or Delete on the Apple IIe and IIc) to erase mistakes while you're typing. When you press Return, *SpeedCalc* displays *T* (for *text*) in the upper-right corner. In this example, the cell isn't long enough to accept all the text, so only the leftmost portion appears in AA2. But even though you can't see it, all of the text is there. Move the cursor to another cell, then move it back to AA2. As soon as you return to AA2, *SpeedCalc* displays all the text in the input buffer area.

Formula data is a mathematical expression or formula. It may be as simple as $2 + 2$ or as complex as your imagination (and mathematical prowess) allows. The first character in a formula must always be an equal sign (=). If you omit this sign, *SpeedCalc* either signals an error or treats the data as text.

The true power of a spreadsheet is that a formula in one cell can refer to another cell. This is easier to demonstrate than to explain. Move the cursor to cell AA3 and type the following line:

=AA1*25.01+@SQR(4)

As soon as you press Return, *SpeedCalc* displays *F* (for *formula*) in the upper-right corner and puts the *result* of the formula (not the formula itself) in AA3. If AA1 contains 123, the value 3078.23 appears in AA3. In plain English, this formula means *multiply the contents of cell AA1 by 25.01 and add the square root of 4*.

Before we examine the formula more closely, here's a quick demonstration of what makes a spreadsheet such a powerful tool. Move the cursor back to AA1 and press Ctrl-R. The command line displays the message *RECALCULATION IS ON*, meaning *SpeedCalc* now automatically recalculates the entire sheet whenever you make a change. Now change the number in AA1 to 456 (simply move to the cell and start typing) and press Return. The new result (11406.56) automatically appears

in cell AA3. We'll explain more about automatic recalculation later.

Note that the referenced cell must contain data that *SpeedCalc* can evaluate—in other words, a number or another formula. If the formula refers to an empty cell or one that contains only text, *SpeedCalc* signals the error by printing *ER-ROR* in the cell containing the incorrect formula.

Mathematical Operators

These symbols can be used as *operators* in a formula:

Operator	Function
+	addition
-	subtraction
*	multiplication
/	division
^	exponentiation
=	equality

One factor that affects formulas is *precedence*, or the order in which mathematical operations are performed. In *SpeedCalc*, formula operators have the same precedence as in ordinary math.

The first operators to be evaluated—those with the highest precedence—are those enclosed in parentheses. Where one set of parentheses encloses another, the expression in the innermost set is evaluated first. The next operators to be evaluated are exponents. Multiplication and division have equal precedence; both operations are lower than exponentiation. Addition and subtraction have the lowest precedence of all. To take one example, *SpeedCalc* evaluates the formula $=5*(8+3*-2)^2-10/+2$ as the value 15, just as in ordinary math. Note how the result is affected by the plus and minus signs before the two 2's.

Functions

Formulas may also include any of these functions:

@ABS()	absolute value
@ATN()	arctangent
@AVE()	average of a block of cells
@COS()	cosine
@EXP()	natural exponent
@INT()	integer
@LOG()	natural logarithm
@SGN()	sign

@SIN()	sine
@SQR()	square root
@SUM()	sum of a block of cells
@TAN()	tangent
PI	value of pi (3.14159265)

All the functions except PI begin with the *at* sign (@) and are followed by parentheses. The parentheses of a function may contain a number or a formula. For example, the formula =@SQR(4) generates the square root of 4. The formula =@SQR(AA1) returns the square root of whatever value cell AA1 contains. Note that the *argument* (the value within parentheses) of the functions @TAN(), @SIN(), and @COS() must be expressed in radians; the result of the function @ATN() is expressed in radians. The function @INT() generates an integer (whole number) by truncating (discarding the fractional part of) a numeric value; note that this is different from rounding.

The function @AVE() calculates the mean average of the values in a block (group) of cells. The function @SUM() calculates the sum of a block. Both functions require you to define the block so that *SpeedCalc* knows which cells to include in the calculation. This is done by putting two cell names separated by a colon in the parentheses. The first cell name defines the upper-left corner of the block, and the second defines the bottom-right corner. For instance, @AVE(AA1:AD20) calculates the average of all the cells from AA1 through AD20. The function @SUM (AA1:AD20) calculates the sum of AA1 through AD20, and so on. An error results if any cell in the block is blank or contains text data.

Editing the Sheet

Editing is a very important spreadsheet function. The simplest way to change what a cell contains is to move to it and start typing. The old data in that cell is replaced by whatever you enter. For instance, to replace the contents of cell AA1 with the number 456, move to that cell, type 456, and press Return or exit with a cursor key. Press Ctrl-B (think of *blank*) to erase what's in the current cell. To erase everything in the sheet, press Ctrl-N (think of *new*). Before carrying out this drastic operation, *SpeedCalc* asks you to confirm it by pressing Y or N.

In some cases, only a minor change is needed. *Edit mode* lets you change the data in a cell without retyping the entire

entry. To activate edit mode, move to the desired cell and press Ctrl-E. In this mode, up and down cursor movement is disabled, and the left/right cursor keys move within the input buffer. Erase unwanted characters with the Esc key (or the Delete key on the Apple IIe and IIc). Typing in edit mode inserts new characters in the line: Everything to the right of the new character moves right one space (unless the buffer is already full). Since the cursor keys have a different function in edit mode, you cannot use them to end the input. Press Return to enter the new data and escape from edit mode.

SpeedCalc displays *ERROR* in a cell when you enter an erroneous formula. Usually, this means you've made a typing error in that cell, or the formula refers to text or an empty cell. A line of asterisks (*****) signals that a number is too large to be printed in the cell. Though these messages appear in the cell area, no data is lost. You may move to the affected cell, view its contents in the input buffer, and make whatever correction is needed.

Recalculation

This feature is the very core of a spreadsheet. As you know, entering or editing a piece of data makes *SpeedCalc* perform a calculation and put the result in the cell under the cursor. In most cases, the new data relates to data in other cells, so you'll ultimately want to recalculate the entire spreadsheet as well. This can be done manually or automatically.

To recalculate the spreadsheet manually, enter an exclamation point (Shift-1). *SpeedCalc* begins at AA1 and recalculates every cell that contains data, placing fresh results wherever needed. If you switch to automatic recalculation mode, *SpeedCalc* automatically recalculates the entire spreadsheet each time you enter new data or edit what exists. When you press Ctrl-R, *SpeedCalc* changes the recalculation status and displays it at the top of the screen. If automatic recalculation was turned off before, it is now on (and vice versa). If you aren't sure which mode you're in, press Open Apple-Ctrl-R; *SpeedCalc* displays the mode without changing it.

Automatic recalculation can be fun to watch in a large spreadsheet: Every time you make a change, new results appear everywhere on the screen. However, the more data your spreadsheet contains, the longer it takes to update the entire sheet. For this reason, you may want to turn off automatic re-

calculation most of the time, recalculating manually whenever you need to view results.

One problem with recalculation arises from the order in which cells are calculated. Because only one cell can be calculated at a time, you must sometimes recalculate the entire spreadsheet two or three times to get correct results in every cell (this is common to all spreadsheet programs). For instance, say you have a formula in AA1 which refers to a formula in AB15. When *SpeedCalc* calculates AA1, it must use the existing data from AB15—which is probably out of date since the formula in AB15 hasn't been recalculated yet. To avoid this problem, you should always recalculate a sheet manually two or three times before printing or saving it to disk.

SpeedCalc offers a number of other features. Before experimenting with them, you should spend some time typing in a hypothetical spreadsheet—perhaps a fictitious yearly budget—to become thoroughly familiar with the basic commands covered so far. Most importantly, create formulas using all the operators in different combinations. Try doing things that you know will cause errors. Then correct the errors in edit mode. It takes a thorough grasp of the fundamentals to get the most out of *SpeedCalc*'s advanced features.

Change Format

The default (normal) format for numeric data is flush right with rounding to two decimal places. In other words, the number is displayed in the rightmost part of the cell, with two numbers after the decimal point. Text and formulas are also displayed flush right. *SpeedCalc* offers several commands for changing cell formats. (Apple II and II+ owners who are using the Esc toggle in place of the Open Apple key should be careful that Esc is not in effect when it's not desired; accidental global changes may be difficult to reverse.)

Change format (Ctrl-F). This command changes the position of data in the cell. When you press Ctrl-F, the *SpeedCalc* command line displays the question *FORMAT: LEFT, CENTER, OR RIGHT JUSTIFY?* Press L, C, or R to move the data to the left, center, or right of the cell.

Change decimal places (#). *SpeedCalc* also lets you change the number of decimal places for any cell. The default number of decimal places is 2, but you may change it to anything from 0 through 15. Press # (Shift-3) to change this

value: *SpeedCalc* prompts you to enter a number from 0 through 15. If you choose zero decimal places, any number in that cell is rounded off to the nearest integer (whole number). If you choose 15, a number in that cell is not rounded off at all—*SpeedCalc* displays it exactly as you entered it or as it was calculated from a formula.

Width (Ctrl-W). The width command changes the width of an entire column of cells. Move the cursor to any cell in the desired column, then press Ctrl-W. When *SpeedCalc* displays the prompt *WIDTH:*, respond with a number from 4 through 36. The entire screen is redrawn to accommodate the new format and may look very different depending on what value you chose. For instance, if you increase a column's width, the rightmost column of the former display may disappear: *SpeedCalc* displays only as many complete columns as it can fit on the screen. If you decrease the width of a column, you may see asterisks where numbers used to be (indicating that the cell is now too small to display the entire number). To get rid of the asterisks, expand the column as necessary.

Global format (Open Apple-Ctrl-F). This is the same as the ordinary format command, but operates globally, changing every cell in the sheet instead of just one.

Global width (Open Apple-Ctrl-W). This is a global version of the width command. Every column in the sheet changes to the designated width.

Macro Editing

After typing in a large spreadsheet, you may decide to make a major change. You may want to add new data somewhere in the middle, delete a section, or move a group of cells from one location to another. *SpeedCalc*'s macro (large-scale) editing commands simplify such operations, affecting an entire block of cells at once. A *block* is simply a group of cells connected in rectangular fashion. You can define it as a single cell, a row or column, or any rectangular area within the spreadsheet.

There are two ways macro commands work: *verbatim* or *relative*. To take a simple example, say that cell AA2 contains the formula `=AA1*5` and you want to move its contents to cell AB2. When this is done in verbatim mode, AB2 contains an exact copy of what was in AA2 (`=AA1*5`). Note that the cell name used in the formula doesn't change—the formula still refers to AA1. If you perform the same operation in relative

mode, the cell name in the formula is adjusted to fit the new location. In this case, AB2 would contain the formula `=AB1*5`. (Apple II and II+ owners using the Esc toggle in place of the Open Apple key should be careful that the toggle is *not* in effect when not desired; accidental relative changes can lead to problems that are difficult to detect and correct.)

Copy (Ctrl-C). The copy command copies a block of cells into a different location without disturbing the original cells. Place the cursor on the upper-left corner of the block you want to copy, then press Ctrl-C. *SpeedCalc* prompts you to move the cursor to the lower-right corner of the block you want to copy. Once the cursor is in place, press Return. Now *SpeedCalc* prompts you to move the cursor to the place where you want to put the block, in other words, the upper-left corner of the new position. Once the cursor is there, press Return again. The new data replaces whatever was contained in the designated cells. Note that if you define an impossible block (for instance, moving the cursor to the upper left of the original position rather than below and to the right), *SpeedCalc* doesn't copy any data. This provides a way to cancel the command if you press Ctrl-C accidentally.

Move (Ctrl-M). This command works like a copy, but it fills the original cells with blanks. Though *SpeedCalc* has no express insert command, you can use this command to make space for new data in the middle of a spreadsheet. Simply move everything below the insertion point down as far as you need.

Because the Return key generates the same character code as Ctrl-M, you may find when you first begin using SpeedCalc that you accidentally invoke the move function by pressing Return when you shouldn't have. To cancel this, simply press Return twice more without moving the cursor.

Relative copy (Open Apple-Ctrl-C). This form of the copy command adjusts the cell names used in formulas within the copied block (see explanation above).

Relative move (Open Apple-Ctrl-M). This is the relative form of the move command. Cell names in formulas are adjusted to reflect the move.

Memory Management

The DOS 3.3 version of *SpeedCalc* makes about 12K (over 12,000 characters) of memory available for data, while the

ProDOS version provides approximately 17K. As noted earlier, *SpeedCalc* lets you spread your data over a much larger number of cells than you can actually fill with data. The extra space is provided to give you full control over the final format of the spreadsheet and to leave some elbow room for move and copy operations.

Because memory is limited, you should carefully keep track of how much is free while using the program. Press Ctrl-A to display the amount of free memory. We suggest limiting your spreadsheets to 1600 cells (equivalent to 40 rows by 40 columns) when using the DOS 3.3 version, or 2000 cells (a 50 × 40 worksheet) when using the ProDOS version. If you've filled nearly all of free memory, you may have to break the spreadsheet into two smaller sheets.

Although *SpeedCalc* checks the amount of available memory and displays an error message if you run out, you should be careful not to exhaust free memory. Any move or copy operation in process will be aborted if sufficient memory is not available.

Disk Operations

SpeedCalc has three disk commands which allow you to save a spreadsheet to disk, load it, and display the disk directory. The directory command is the simplest to use—simply press Ctrl-D. The spreadsheet disappears, and a directory of the disk in drive 1 is displayed. Press Return to return to the spreadsheet.

To save a spreadsheet to disk, press Ctrl-S. *SpeedCalc* prints *SAVE:* on the command line, followed by an underline cursor. Enter a valid Apple filename and press Return. (If you change your mind and decide not to save anything, press Return without typing a filename.) If no disk error occurs while the spreadsheet is being saved, *SpeedCalc* displays *NO ERRORS* in the command line and returns you to command mode. If there was an error, you'll hear a beep and see the message *I/O ERROR* in the command line.

To load a saved file from disk, press Ctrl-L. Again, you can cancel the operation by pressing Return without entering a filename. *SpeedCalc* prompts you to enter the filename and displays the error status when the operation is complete.

When saving or loading *SpeedCalc* files with ProDOS, you must specify the prefix along with the name. If you don't want to type the prefix every time you enter a filename, simply call

up a directory for the disk you want to use to save or load. This automatically sets the prefix to match the current disk, relieving you of the need to enter it with every name.

Printing

SpeedCalc lets you print data to three different devices: to the screen for previewing output, to a printer for permanent documentation, or to a disk file for integrating the data with a *SpeedScript* document.

To print a hardcopy of the spreadsheet to a printer in slot 1, press Ctrl-P. *Before using this command, you must position the cursor below and to the right of the block of cells you wish to print.* The upper-left corner of the printout starts at cell AA1.

To send output to a printer with a slot number other than 1, or to the screen or a disk, first position the cursor in the lower-right corner of the block you want to print. Then press Open Apple-Ctrl-P (toggle Esc on the Apple II and II+). *SpeedCalc* asks if you want to print to the screen, to disk, or to the printer. Press S to preview output on the screen, D to print to disk, or P to select printer output. Pressing any other key cancels the command.

If you select the P option after pressing Open Apple-Ctrl-P, *SpeedCalc* asks you to specify a slot number by pressing one of the number keys from 1 through 7. This permits you to use a printer in any of those slots. If you change your mind at any point during this process, press Return without entering anything; *SpeedCalc* returns you to command mode.

You can also print *SpeedCalc* data to a disk file for use in a *SpeedScript* document. Select the D option after pressing Open Apple-Ctrl-P, then enter a filename. The data is saved as a disk file of that name. Note that *printing* to disk creates a different type of file than does *saving* to disk, and *SpeedCalc* cannot reload files in the print format. You should *save* files you wish to reload into *SpeedCalc* and *print* files you wish to convert for *SpeedScript*. Unlike the *SpeedCalc* save and load commands, no error messages are provided if the spreadsheet cannot be printed to disk. Thus, you must insure that the drive contains a write-enabled disk with sufficient space to hold the printed spreadsheet before you attempt to print to disk.

***SpeedScript* File Converter**

SpeedCalc sends data to the printer in simple, plain-vanilla form. That may be fine for personal use, but if you're creating a document for others to view, you may want special features such as boldface and underlining. Since Apple *SpeedScript*—*COMPUTE!*'s popular word processor—already offers a way to access these features (and many more), no attempt has been made to include them in *SpeedCalc*. All that's needed is a simple program to convert *SpeedCalc* files into a form that *SpeedScript* can load. Then you can edit the file with *SpeedScript* as you would any other document—inserting printer control codes, reformatting the text, merging it with other text, and the like. The *SpeedScript* File Converter program published with *SpeedScript* (both when it appeared in *COMPUTE!* magazine and in its *COMPUTE!* book form) makes it easy to perform the conversion. Assuming that you have a copy of *SpeedScript* for the Apple and the File Converter program, here are the steps to follow to convert a *SpeedCalc* file for *SpeedScript*:

- After creating a spreadsheet with *SpeedCalc*, print it to disk as described above.
- Exit *SpeedCalc*, then load and run the *SpeedScript* "File Converter" program. The program prompts you to enter the name of the *SpeedCalc* file you printed to disk. Then it asks you to enter the name of the *SpeedScript* file you want to create (of course, this name should be different from the first). The File Converter then constructs a *SpeedScript*-loadable disk file from the *SpeedCalc* file.
- After the File Converter is finished, load and BRUN *SpeedScript*, then load the new *SpeedScript* file as you would any *SpeedScript* document. The data appears on the screen, ready to be edited in any way you wish.

SpeedCalc Commands

Command	Action
Ctrl-A	Available memory check
Ctrl-B	Blank (erase) current cell
Ctrl-C	Copy block verbatim
Ctrl-D	Disk directory
Ctrl-E	Edit current cell
Ctrl-F	Change cell format
Ctrl-G	Goto selected cell
Ctrl-L	Load <i>SpeedCalc</i> file
Ctrl-M	Move block verbatim
Ctrl-N	New (erase entire sheet)
Ctrl-P	Print file on printer
Ctrl-R	Turn recalculation on/off
Ctrl-S	Save <i>SpeedCalc</i> file
Ctrl-W	Change column width
Ctrl-X	Exit <i>SpeedCalc</i>
Ctrl-@	Home cursor
Open Apple-Ctrl-C	Copy block relative
Open Apple-Ctrl-M	Move block relative
Open Apple-Ctrl-P	Print to screen, disk, or printer
Open Apple-Ctrl-R	Check recalculation status
Open Apple-Ctrl-W	Change width of all columns
! (Shift-1)	Recalculate sheet
# (Shift-3)	Change decimal places

Note: The Apple II and II+ have no Open Apple key, so Esc must be used as an Open Apple toggle. Pressing Esc once makes all following keypresses behave as if Open Apple were pressed. Press Esc again to turn off the Open Apple toggle.

Program 1. Apple SpeedCalc for DOS 3.3

For mistake-proof program entry, use "AppleMLX" (Appendix C) to type in this program.

START ADDRESS: 07FA
END ADDRESS: 24F9

```

07FA: 20 65 D6 4C D2 D7 00 0A 12
0802: 08 0A 00 A5 AB 33 30 00 7D
080A: 14 08 14 00 8C 32 30 38 6E
0812: 33 00 1E 08 1E 00 8C 32 3C
081A: 30 38 30 00 00 00 4C 4D 3C
0822: 22 20 58 FC AD 61 C0 8D 28
082A: 59 25 A9 00 8D F2 03 A9 4D
0832: 09 8D F3 03 49 A5 8D F4 C9
083A: 03 A9 FD 85 39 85 37 A9 46
0842: 1B 85 38 A9 F0 85 36 A9 96
084A: 25 18 69 01 8D F0 24 18 C0
0852: 69 4F 85 6C A9 00 8D EF BA

```



```

085A: 24 8D F1 24 85 6B 8D 69 BE
0862: 22 85 FF 8D 58 25 A9 A5 0E
086A: 8D F2 24 A9 09 20 61 09 B1
0872: 20 D9 0A A9 23 A0 46 20 2D
087A: 3E 09 20 88 0D 20 25 09 B4
0882: 48 20 7C 09 68 AE AC 08 3E
088A: DD AC 08 F0 0A CA D0 F8 DA
0892: C9 20 90 E6 4C 37 0C CA 32
089A: 8A 0A AA A9 08 48 A9 7B 92
08A2: 48 BD D3 08 48 BD D2 08 28
08AA: 48 60 17 0E 00 17 06 07 2A
08B2: 10 03 13 0C 18 0A 0B 15 C2
08BA: 08 02 05 21 01 12 04 0D 67
08C2: 1B 23 0D 31 32 33 34 35 D9
08CA: 36 37 38 39 30 2B 2D 2E 15
08D2: C4 0A DB 11 13 10 A8 0C 8A
08DA: 4E 11 32 14 ED 15 A2 19 FF
08E2: 40 1A D5 1E DD 10 F6 10 63
08EA: 0D 11 37 11 94 1C DB 1C A6
08F2: 08 1C 03 1E A7 1C CC 1B B2
08FA: C8 15 08 09 ED 0C 20 58 7E
0902: FC 20 22 0B 4C 75 08 AD 85
090A: 58 25 49 FF 8D 58 25 60 33
0912: 2C 00 C0 10 0B AD 00 C0 23
091A: 8D 10 C0 29 7F C9 FF 60 25
0922: A9 00 60 A5 FF F0 07 48 89
092A: A9 00 85 FF 68 60 20 12 D8
0932: 09 F0 FB 60 20 F2 EB A5 D4
093A: A0 A4 A1 60 85 FC 84 FB 25
0942: 20 6F 09 20 80 FE A9 00 B6
094A: 85 28 85 24 85 25 A9 04 34
0952: 85 29 A0 00 B1 FB F0 06 EA
095A: 20 ED FD C8 D0 F6 60 A2 0A
0962: 32 9D F6 24 CA D0 FA A9 4F
096A: 28 8D 29 25 60 A0 00 A9 9A
0972: 20 99 00 04 C8 C0 28 D0 A5
097A: F6 60 AD 01 04 C9 10 D0 1E
0982: 0A AD 0A 04 C9 02 F0 03 C1
098A: 4C 94 09 A9 23 A0 3C 20 D7
0992: 3E 09 38 20 C7 1F 90 03 ED
099A: 4C 32 0F 4C 40 0F 09 80 D6
09A2: 8D 80 02 A9 3C 8D 81 02 93
09AA: A2 76 A9 A0 9D 81 02 CA AC
09B2: D0 FB A0 01 D0 02 A0 00 5F
09BA: B9 80 02 8D 3C 25 A9 DF 8C
09C2: 99 80 02 20 AB 0A 20 12 DB
09CA: 09 D0 16 EE 3B 25 10 08 DD
09D2: A9 DF 99 80 02 4C C5 09 C2
09DA: AD 3C 25 99 80 02 4C C5 7B
09E2: 09 09 80 8D 3B 25 AD 3C AA

```

```

09EA: 25 99 80 02 AD 3B 25 AE 79
09F2: 95 0A DD 95 0A F0 2C CA 9E
09FA: D0 F8 C9 A0 90 BA 8C 3C BB
0A02: 25 CE 3C 25 A2 77 BD 80 25
0A0A: 02 C9 3C F0 AB CA BD 80 AC
0A12: 02 9D 81 02 CA EC 3C 25 86
0A1A: D0 F4 AD 3B 25 99 80 02 CF
0A22: C8 D0 95 CA 8A 0A AA BD BD
0A2A: 9E 0A 48 BD 9D 0A 48 60 FA
0A32: A0 00 B9 80 02 C9 3C F0 76
0A3A: 08 29 7F 99 00 03 C8 D0 94
0A42: F1 A9 00 99 00 03 8C 2C A4
0A4A: 25 60 AD 6A 22 F0 20 C0 3B
0A52: 00 F0 01 88 4C BA 09 AD 58
0A5A: 6A 22 F0 13 B9 80 02 C9 19
0A62: 3C F0 F1 C8 4C BA 09 AD A8
0A6A: 6A 22 F0 03 4C BA 09 AD 97
0A72: 3B 25 29 7F 85 FF 4C 32 81
0A7A: 0A C0 00 F0 D7 88 98 AA 8F
0A82: BD 81 02 9D 80 02 E8 C9 97
0A8A: 3C D0 F5 A9 A0 9D 80 02 C8
0A92: 4C BA 09 07 8D 9B 8A 8B 88
0A9A: 88 95 FF 31 0A 7A 0A 68 22
0AA2: 0A 68 0A 4B 0A 58 0A 7A 0C
0AAA: 0A A2 00 BD 80 02 9D 80 10
0AB2: 04 BD A8 02 9D 00 05 BD 22
0ABA: D0 02 9D 80 05 E8 E0 28 29
0AC2: D0 E9 60 A9 23 A0 60 20 DC
0ACA: 3E 09 20 25 09 C9 59 D0 89
0AD2: 03 20 D9 0A 4C 7C 09 20 D2
0ADA: FA 0A A9 09 20 61 09 20 6D
0AE2: 22 0B 20 88 0D A9 2C 8D 4C
0AEA: B4 22 A9 00 8D B3 22 A5 3C
0AF2: 6B 85 6F A5 6C 85 70 60 21
0AFA: AD EF 24 85 FB AD F0 24 5B
0B02: 85 FC A0 00 98 91 FB C8 F9
0B0A: D0 FB E6 FC A6 FC EC F2 2A
0B12: 24 D0 F2 A9 01 8D F4 24 B3
0B1A: 8D F5 24 85 1D 85 1E 60 EC
0B22: 20 28 0B 4C B0 0B A0 05 70
0B2A: 8C 3B 25 B9 83 22 85 28 6D
0B32: B9 6B 22 85 29 A0 00 AE 17
0B3A: F3 24 A9 00 8D 29 25 8D 72
0B42: 2A 25 F8 AD 29 25 18 69 28
0B4A: 01 8D 29 25 AD 2A 25 69 85
0B52: 00 8D 2A 25 CA D0 EC D8 AF
0B5A: A2 00 20 8D 0B F8 AD 29 5F
0B62: 25 18 69 01 8D 29 25 AD 57
0B6A: 2A 25 69 00 8D 2A 25 D8 44
0B72: EE 3B 25 AC 3B 25 B9 83 A3

```

```

0B7A: 22 85 28 B9 6B 22 85 29 BB
0B82: A0 00 E8 E0 12 D0 D3 20 AF
0B8A: 8D 0B 60 AD 2A 25 18 69 90
0B92: 30 91 28 C8 AD 29 25 29 3C
0B9A: F0 4A 4A 4A 4A 18 69 30 5F
0BA2: 91 28 C8 AD 29 25 29 0F BE
0BAA: 18 69 30 91 28 60 A0 04 4E
0BB2: B9 83 22 85 28 B9 6B 22 44
0BBA: 85 29 A0 00 A9 20 91 28 0B
0BC2: C8 91 28 C8 91 28 C8 AE A0
0BCA: F4 24 A9 00 8D F3 24 BD DB
0BD2: F6 24 8E 29 25 4A 69 00 F6
0BDA: AA CA A9 20 91 28 C8 CA B9
0BE2: D0 FA AD 29 25 0A AA BD CC
0BEA: B5 22 29 3F 91 28 C8 BD F9
0BF2: B6 22 29 3F 91 28 C8 AE 73
0BFA: 29 25 BD F6 24 4A AA CA 80
0C02: CA A9 20 91 28 C8 CA 10 11
0C0A: FA AE 29 25 BD F6 24 18 EC
0C12: 6D F3 24 8D F3 24 E8 BD FA
0C1A: F6 24 18 6D F3 24 C9 25 79
0C22: 90 AD CA 8E 32 25 A9 20 C9
0C2A: C0 28 D0 01 60 91 28 C8 39
0C32: C0 28 D0 F9 60 20 A0 09 3C
0C3A: AD 00 03 F0 3F C9 3D F0 25
0C42: 26 AE C4 08 DD C4 08 F0 35
0C4A: 07 CA D0 F8 A9 01 D0 19 4E
0C52: AD 2C 25 C9 25 B0 25 A0 64
0C5A: 00 A9 03 20 81 0C 20 B7 73
0C62: 00 D0 E9 A9 00 F0 02 A9 F7
0C6A: 02 8D 2B 25 AD B4 22 8D B0
0C72: 2D 25 18 20 C7 1F 20 27 91
0C7A: 20 20 03 1C 4C 7C 08 85 B6
0C82: B9 84 B8 20 B7 00 4C 4A 52
0C8A: EC A2 32 A9 00 8D 30 25 6E
0C92: BD F6 24 18 6D 38 25 8D 71
0C9A: 38 25 C9 25 B0 03 CA D0 9B
0CA2: EF E8 E8 8E 3C 25 60 A9 D3
0CAA: 00 2C 59 25 30 03 AD 61 95
0CB2: C0 0D 58 25 8D 57 25 A0 80
0CBA: A5 A9 23 20 3E 09 20 25 F1
0CC2: 09 C9 4C F0 0F C9 43 F0 81
0CCA: 0F C9 52 F0 03 4C 85 0D 97
0CD2: A2 0C D0 06 A2 08 D0 02 92
0CDA: A2 04 AD B4 22 29 F0 8D 6B
0CE2: 3B 25 8A 0D 3B 25 8D 3B C8
0CEA: 25 4C 2F 0D A9 00 2C 59 5E
0CF2: 25 30 03 AD 61 C0 0D 58 65
0CFA: 25 8D 57 25 A0 CE A9 23 FC
0D02: 20 3E 09 20 76 10 F0 7B 30

```



```

0D0A: A0 00 A9 02 20 81 0C 20 09
0D12: 36 09 C9 00 D0 6D C0 10 90
0D1A: B0 69 AD B4 22 29 0C 8D 43
0D22: 3B 25 98 0A 0A 0A 0A 0D 70
0D2A: 3B 25 8D 3B 25 AD 57 25 44
0D32: 10 41 AD 3B 25 8D B4 22 F8
0D3A: AD EF 24 85 1B AD F0 24 99
0D42: 85 1C A0 01 B1 1B F0 11 37
0D4A: 85 1A 88 B1 1B 85 19 B1 AC
0D52: 19 29 03 0D B4 22 91 19 DE
0D5A: C8 A5 1B 18 69 02 85 1B A0
0D62: A5 1C 69 00 85 1C A5 1C 87
0D6A: C5 6C D0 D8 38 20 C7 1F 1B
0D72: 4C 85 0D 38 20 C7 1F 90 28
0D7A: 0A A0 00 AD 3B 25 0D 2B 50
0D82: 25 91 19 4C 7C 09 A5 1D EB
0D8A: 8D 30 25 A5 1E 8D 31 25 25
0D92: A9 03 8D F3 24 AE F4 24 1D
0D9A: 86 1D AC F5 24 84 1E 98 3C
0DA2: 18 69 13 8D 2E 25 BD F6 D6
0DAA: 24 8D 38 25 A9 FF EC 30 EA
0DB2: 25 D0 07 CC 31 25 D0 02 03
0DBA: A9 3F 8D 33 25 98 18 69 83
0DC2: 05 38 ED F5 24 A8 B9 6B 2D
0DCA: 22 85 29 B9 83 22 85 28 EF
0DD2: 38 20 C7 1F B0 05 A9 A0 89
0DDA: 4C 67 0E AD 2B 25 F0 70 D1
0DE2: C9 02 F0 6C AD 38 25 38 18
0DEA: ED 2C 25 AA E8 30 32 E8 AB
0DF2: AD 2D 25 29 0C C9 08 F0 EE
0DFA: 28 B0 05 8A 4A F0 22 AA A3
0E02: 8E 34 25 A9 A0 2D 33 25 F6
0E0A: AC F3 24 91 28 C8 CA D0 E1
0E12: FA 8C 35 25 AD 38 25 38 98
0E1A: ED 34 25 AA A0 02 4C 2E 5D
0E22: 0E AE 38 25 AD F3 24 8D 5D
0E2A: 35 25 A0 02 B1 19 8C 34 9D
0E32: 25 AC 35 25 09 80 2D 33 DC
0E3A: 25 91 28 AC 34 25 EE 35 66
0E42: 25 CA F0 09 C8 CC 2C 25 49
0E4A: D0 E2 20 A9 0E 4C 76 0E C2
0E52: 20 4E 0F AE 2C 25 CA CA 35
0E5A: CA EC 38 25 B0 03 4C E6 81
0E62: 0D A9 2A 09 80 2D 33 25 89
0E6A: AC F3 24 AE 38 25 91 28 EA
0E72: C8 CA D0 FA A4 1E A6 1D 77
0E7A: C8 CC 2E 25 F0 05 84 1E 09
0E82: 4C A8 0D AC F5 24 84 1E C2
0E8A: AD 38 25 18 6D F3 24 8D C2
0E92: F3 24 E8 86 1D E0 33 F0 FA

```

```

0E9A: 27 BD F6 24 18 6D F3 24 5D
0EA2: C9 28 B0 1C 4C A8 0D E0 85
0EAA: 00 F0 14 AD F3 24 18 6D 2E
0EB2: 38 25 A8 88 A9 A0 2D 33 2F
0EBA: 25 91 28 88 CA D0 FA 60 4B
0EC2: A9 28 38 ED F3 24 8D 38 27
0ECA: 25 A0 05 84 1E B9 6B 22 5B
0ED2: 85 29 B9 83 22 85 28 AC 8F
0EDA: F3 24 AE 38 25 A9 A0 91 F5
0EE2: 28 C8 CA D0 FA E6 1E A4 FF
0EEA: 1E C0 18 D0 E0 AD 30 25 99
0EF2: 85 1D AD 31 25 85 1E A0 FD
0EFA: 00 A9 A0 99 80 02 C8 C0 8D
0F02: 78 D0 F8 38 20 C7 1F 90 22
0F0A: 35 A0 02 A2 00 AD 2B 25 87
0F12: C9 02 D0 09 AC 2C 25 B1 52
0F1A: 19 8D 2C 25 C8 B1 19 09 48
0F22: 80 9D 80 02 E8 C8 CC 2C 48
0F2A: 25 D0 F2 A9 3C 9D 80 02 63
0F32: AE 2B 25 BD 80 22 29 3F 92
0F3A: 8D 27 04 4C AB 0A A9 20 27
0F42: 8D 27 04 A9 3C 8D 80 02 27
0F4A: 20 AB 0A 60 A9 20 8D 00 93
0F52: 02 A0 02 B1 19 C9 2A F0 2A
0F5A: F2 AD 2D 25 4A 4A 4A 4A AF
0F62: 8D 36 25 A2 FF C9 0F F0 D9
0F6A: E2 B1 19 C9 2E D0 09 AE 9B
0F72: 36 25 F0 10 E8 8E 00 02 97
0F7A: 99 FF 01 C8 CC 2C 25 F0 64
0F82: 03 CA D0 E5 AD 36 25 F0 CE
0F8A: 1E E0 00 F0 1A AD 00 02 88
0F92: C9 20 D0 0A A9 2E 99 FF 91
0F9A: 01 C8 AE 36 25 E8 A9 30 F4
0FA2: 99 FF 01 C8 CA D0 F9 A9 71
0FAA: 20 8D 00 02 CC 2C 25 F0 AE
0FB2: 0C B0 3F B1 19 C9 2E F0 43
0FBA: 08 C9 35 B0 0C C8 4C F4 12
0FC2: 0F C8 B1 19 C9 35 90 2A D0
0FCA: 88 98 C8 AA CA CA BD 00 14
0FD2: 02 C9 2E F0 0B 90 0C C9 B5
0FDA: 39 D0 14 A9 30 9D 00 02 E0
0FE2: CA 10 EB CA 9D 00 02 E8 6E
0FEA: A9 31 9D 00 02 D0 03 FE 36
0FF2: 00 02 88 8C 2C 25 AD 00 BC
0FFA: 02 C9 20 D0 09 A9 01 85 14
1002: 1A A9 FF 85 19 60 A9 01 90
100A: 85 1A A9 FE 85 19 EE 2C 33
1012: 25 60 A9 00 2C 59 25 30 53
101A: 03 AD 61 C0 0D 58 25 8D 01
1022: 57 25 A9 23 A0 79 20 3E 08

```

102A: 09 20 76 10 A0 00 A9 02 01
1032: 20 81 0C 20 36 09 C9 00 AF
103A: D0 33 C0 04 90 2F C0 25 CF
1042: B0 2B A5 1D 8D F4 24 AD 42
104A: 57 25 10 07 98 20 61 09 E2
1052: 4C 5B 10 98 A6 1D 9D F6 D6
105A: 24 20 8B 0C A5 1D CD 3C 40
1062: 25 90 07 AC 3C 25 88 8C F8
106A: F4 24 20 B0 0B 4C 7C 09 AB
1072: A9 01 D0 02 A9 00 8D 37 81
107A: 25 A0 00 A9 1F 20 ED FD 43
1082: A9 88 20 ED FD 20 25 09 40
108A: C9 0D F0 3F C9 08 F0 26 5B
1092: C9 7F F0 22 C9 20 90 ED 95
109A: AE 37 25 D0 08 C9 30 90 E9
10A2: E4 C9 3A B0 E0 A6 24 E0 C4
10AA: 26 F0 DA 99 00 02 09 80 A9
10B2: 20 ED FD C8 D0 C5 C0 00 C9
10BA: F0 CB A9 A0 20 ED FD A9 E3
10C2: 88 20 ED FD 20 ED FD 88 0A
10CA: 4C 7D 10 A9 A0 20 ED FD 6C
10D2: A9 00 99 00 02 8C 36 25 CE
10DA: AD 00 02 60 A5 1E C9 C8 1A
10E2: F0 12 E6 1E AD F5 24 18 64
10EA: 69 12 C5 1E B0 06 EE F5 50
10F2: 24 20 28 0B 60 A5 1E C9 82
10FA: 01 F0 10 C6 1E AC F5 24 F9
1102: 88 C4 1E 90 06 CE F5 24 E1
110A: 20 28 0B 60 A5 1D C9 32 15
1112: F0 23 E6 1D AC 32 25 C4 61
111A: 1D B0 1A EE F4 24 AE F4 B3
1122: 24 A9 00 18 7D F6 24 E8 3B
112A: C9 25 90 F7 CA CA E4 1D 74
1132: 90 E9 20 B0 0B 60 A5 1D 68
113A: C9 01 F0 10 C6 1D AC F4 99
1142: 24 88 C4 1D 90 06 CE F4 32
114A: 24 20 B0 0B 60 A9 23 A0 DD
1152: 80 20 3E 09 20 72 10 A9 A9
115A: 01 85 B9 A9 FF 85 B8 20 D7
1162: B1 00 90 4E 38 E9 41 30 70
116A: 49 F0 06 C9 02 B0 43 A9 CD
1172: 1A 8D 3B 25 20 B1 00 90 17
117A: 39 38 E9 40 30 34 F0 32 EE
1182: C9 1B B0 2E 18 6D 3B 25 5B
118A: C9 33 B0 26 8D 3B 25 20 9A
1192: B1 00 B0 1E 20 4A EC 20 A9
119A: 36 09 C9 00 D0 14 C0 00 AB
11A2: F0 10 C0 C9 B0 0C C0 B7 E4
11AA: 90 0B A9 B6 8D F5 24 4C 51
11B2: BA 11 4C 7C 09 8C F5 24 52


```

11BA: 84 1E 20 8B 0C AD 3B 25 16
11C2: CD 3C 25 90 0A AC 3C 25 29
11CA: 88 8C F4 24 4C D4 11 8D 9A
11D2: F4 24 85 1D 20 22 0B 4C E6
11DA: 7C 09 AD F4 24 C5 1D D0 C5
11E2: 17 AD F5 24 C5 1E D0 10 55
11EA: A9 01 8D F4 24 85 1D 8D 22
11F2: F5 24 85 1E 20 22 0B 60 AB
11FA: AD F4 24 85 1D AD F5 24 BD
1202: 85 1E 60 20 B1 00 8D 4F 76
120A: 25 20 B1 00 8D 50 25 20 17
1212: B1 00 8D 51 25 20 B1 00 E2
121A: C9 28 F0 03 4C 4D 22 AE 06
1222: 6A 12 AD 4F 25 DD 6A 12 32
122A: F0 06 CA D0 F5 4C 4D 22 4C
1232: AD 50 25 DD 76 12 F0 02 A3
123A: D0 F0 AD 51 25 DD 82 12 85
1242: D0 E8 8E 29 25 E0 0B B0 E0
124A: 0C 8A 48 A9 00 48 4C 22 96
1252: 21 68 8D 29 25 20 B1 00 72
125A: AE 29 25 CA 8A 0A AA BD 01
1262: 90 12 48 BD 8F 12 48 60 ED
126A: 0C 41 41 43 45 49 4C 53 7C
1272: 53 53 54 53 41 42 54 4F DF
127A: 58 4E 4F 47 49 51 41 55 24
1282: 56 53 4E 53 50 54 47 4E 56
128A: 4E 52 4E 4D 45 AE EB 9D 63
1292: F0 E9 EF 08 EF 22 EC 40 4A
129A: E9 8F EB F0 EF 8C EE 39 EC
12A2: F0 A9 13 11 14 20 64 13 1A
12AA: 8E 52 25 8C 54 25 20 B7 47
12B2: 00 C9 3A D0 3F 20 B1 00 7B
12BA: 20 64 13 8E 53 25 8C 55 F0
12C2: 25 20 B7 00 C9 29 D0 2C 39
12CA: 20 B1 00 AE 52 25 CA EC FF
12D2: 53 25 90 03 4C 4D 22 AC B4
12DA: 54 25 88 CC 55 25 90 03 B3
12E2: 4C 4D 22 E8 C8 A5 1D 8D F7
12EA: 39 25 A5 1E 8D 3A 25 86 B1
12F2: 1D 84 1E 60 4C 4D 22 18 84
12FA: 20 C7 1F 90 42 A0 00 B1 54
1302: 19 29 03 C9 01 F0 38 C8 01
130A: B1 19 8D 3C 25 A2 00 C8 41
1312: B1 19 9D 00 02 E8 C8 CC 1D
131A: 3C 25 D0 F4 A5 B8 48 A5 57
1322: B9 48 A9 00 9D 00 02 A9 07
132A: 02 A0 00 20 B1 0C 68 85 0E
1332: B9 68 85 B8 A5 1D CD 53 1C
133A: 25 F0 15 E6 1D 18 60 AD F7
1342: 39 25 85 1D AD 3A 25 85 F6

```

```

134A: 1E 18 20 C7 1F 4C 4D 22 EC
1352: AD 52 25 85 1D A5 1E CD 6A
135A: 55 25 F0 04 E6 1E 18 60 13
1362: 38 60 A2 00 20 B7 00 C9 BA
136A: 41 F0 06 C9 42 D0 D0 A2 64
1372: 1A 8E 3B 25 20 B1 00 C9 94
137A: 41 90 C4 C9 5B B0 C0 38 F1
1382: E9 40 18 6D 3B 25 C9 33 BC
138A: B0 B5 8D 3B 25 20 B1 00 E8
1392: B0 AD 20 4A EC 20 36 09 82
139A: C9 00 D0 A3 C0 00 F0 9F 81
13A2: C0 C9 B0 9B AE 3B 25 60 78
13AA: A9 01 8D 29 25 A9 00 8D 87
13B2: 2A 25 20 A7 12 20 F9 12 CC
13BA: B0 47 20 72 EB A5 A2 48 B9
13C2: A5 A1 48 A5 A0 48 A5 9F 98
13CA: 48 A5 9E 48 A5 9D 48 EE F9
13D2: 29 25 D0 03 EE 2A 25 20 AB
13DA: F9 12 08 68 8D 3B 25 68 16
13E2: 85 A5 68 85 A6 68 85 A7 24
13EA: 68 85 AB 68 85 A9 68 85 6B
13F2: AA 45 A2 85 AB A5 9D 20 BB
13FA: C1 E7 AD 3B 25 48 28 90 90
1402: B9 AD 39 25 85 1D AD 3A 22
140A: 25 85 1E 18 20 C7 1F 60 2A
1412: 20 AA 13 A2 06 B5 9C 95 57
141A: A4 CA D0 F9 AD 2A 25 AC 0E
1422: 29 25 20 F2 E2 A5 AA 45 A3
142A: A2 85 AB A5 9D 20 F3 21 4B
1432: 60 20 58 FC A9 01 8D 56 30
143A: 25 A9 00 2C 59 25 30 03 E4
1442: AD 61 C0 0D 58 25 30 03 3D
144A: 4C A9 14 A9 24 A0 91 20 07
1452: 3E 09 20 25 09 C9 53 F0 39
145A: 0B C9 44 F0 0E C9 50 F0 3B
1462: 28 4C B2 15 A9 03 8D 56 24
146A: 25 D0 3C A9 00 8D 56 25 83
1472: A0 B5 A9 24 20 3E 09 20 FB
147A: 72 10 A9 00 AA 20 0A 1B 1A
1482: F0 25 C9 06 F0 21 4C A5 50
148A: 15 A9 24 A0 8A 20 3E 09 90
1492: 20 25 09 38 E9 30 C9 00 5C
149A: B0 03 4C B2 15 C9 08 90 01
14A2: 03 4C B2 15 8D 56 25 A9 C0
14AA: 24 A0 7E 20 3E 09 20 84 B9
14B2: FE AD 56 25 F0 14 C9 03 51
14BA: D0 0D AD 05 C3 18 6D 07 F4
14C2: C3 C9 50 D0 05 A9 03 20 4B
14CA: 95 FE A5 1D 8D 53 25 8D 95
14D2: 30 25 A5 1E 8D 55 25 8D 8C

```

```

14DA: 31 25 A9 01 85 1D 85 1E F3
14E2: A9 8D 20 BB 15 A6 1D BD 3E
14EA: F6 24 8D 38 25 AA A9 00 F3
14F2: 9D 00 03 CA A9 20 9D 00 FF
14FA: 03 CA 10 FA 38 20 C7 1F F9
1502: 90 58 AD 2B 25 C9 01 D0 16
150A: 23 AD 38 25 38 ED 2C 25 81
1512: AA E8 30 14 E8 AD 2D 25 90
151A: 29 0C C9 08 F0 0A B0 27 CD
1522: 8A 4A F0 04 AA 4C 49 15 B0
152A: A2 00 F0 1B 20 4E 0F AE 7C
1532: 2C 25 CA CA CA EC 38 25 61
153A: 90 CF AE 38 25 A9 2A 9D BB
1542: FF 02 CA D0 FA F0 13 A0 B5
154A: 02 B1 19 9D 00 03 E8 C8 85
1552: EC 38 25 F0 05 CC 2C 25 8D
155A: D0 EF A2 00 BD 00 03 F0 22
1562: 08 09 80 20 BB 15 E8 D0 B9
156A: F3 A5 1D CD 53 25 F0 05 8E
1572: E6 1D 4C E7 14 A5 1E CD A0
157A: 55 25 F0 0E E6 1E A9 01 9B
1582: 85 1D A9 8D 20 BB 15 4C 2B
158A: E7 14 A9 8D 20 BB 15 AD 83
1592: 56 25 C9 03 D0 03 20 25 92
159A: 09 A9 00 20 95 FE AD 56 10
15A2: 25 D0 03 20 52 1B AD 30 80
15AA: 25 85 1D AD 31 25 85 1E 8E
15B2: 20 58 FC 20 22 0B 4C 7C F6
15BA: 09 48 AD 56 25 F0 04 68 F3
15C2: 4C ED FD 68 4C 95 1B A9 6D
15CA: 00 2C 59 25 30 03 AD 61 C7
15D2: C0 0D 58 25 8D 3F 25 A9 5B
15DA: 00 8D 40 25 A5 1D 8D 41 C0
15E2: 25 A5 1E 8D 42 25 4C 0D F1
15EA: 16 4C 7C 09 A9 00 2C 59 52
15F2: 25 30 03 AD 61 C0 0D 58 77
15FA: 25 8D 3F 25 A9 01 8D 40 02
1602: 25 A5 1D 8D 41 25 A5 1E AE
160A: 8D 42 25 20 48 16 AD 30 5A
1612: 25 8D 45 25 AD 31 25 8D 39
161A: 46 25 20 52 16 AE 41 25 EE
1622: CA EC 45 25 B0 13 AE 42 5B
162A: 25 CA EC 46 25 B0 0A A9 47
1632: 23 A0 FA 20 3E 09 20 44 14
163A: 18 AD 43 25 85 1D AD 44 D8
1642: 25 85 1E 4C 7C 09 A9 24 6A
164A: A0 59 20 3E 09 4C 59 16 47
1652: A9 24 A0 31 20 3E 09 20 AF
165A: 88 0D 20 25 09 AE 8D 16 98
1662: DD 8D 16 F0 06 CA D0 F8 A8

```



```

166A: 4C 59 16 CA 8A 0A AA A9 FD
1672: 16 48 A9 58 48 BD 95 16 F0
167A: 48 BD 94 16 48 60 68 68 2B
1682: A5 1D 8D 43 25 A5 1E 8D 3B
168A: 44 25 60 06 00 0B 0A 0B D6
1692: 15 0D DB 11 F6 10 DD 10 DC
169A: 37 11 0D 11 7F 16 AD 49 52
16A2: 25 C9 33 B0 5B AD 4A 25 90
16AA: C9 C9 B0 54 AD 47 25 85 E3
16B2: 1D AD 48 25 85 1E 38 20 69
16BA: C7 1F 90 45 A0 02 AD 2B 8C
16C2: 25 C9 02 D0 09 AC 2C 25 B9
16CA: B1 19 8D 2C 25 C8 A2 00 1C
16D2: B1 19 9D 00 03 E8 C8 CC EB
16DA: 2C 25 D0 F4 A9 00 9D 00 58
16E2: 03 8E 2C 25 20 25 17 AD 7D
16EA: 40 25 D0 03 20 13 17 AD F3
16F2: 49 25 85 1D AD 4A 25 85 F5
16FA: 1E 18 20 C7 1F 20 27 20 A4
1702: 60 AD 49 25 85 1D AD 4A 8D
170A: 25 85 1E 18 20 C7 1F 90 60
1712: EF 20 33 1E 18 20 C7 1F 78
171A: A9 00 A8 91 1B C8 91 1B 85
1722: 4C 02 17 AD 3F 25 30 01 A4
172A: 60 AD 2B 25 C9 02 F0 01 E4
1732: 60 AD 49 25 38 ED 47 25 A4
173A: 8D 4D 25 AD 4A 25 38 ED 47
1742: 48 25 8D 4E 25 A2 00 8E B6
174A: 2A 25 BD 00 03 9D 80 02 20
1752: E8 EC 2C 25 D0 F4 A9 00 B5
175A: 9D 80 02 A9 80 85 B8 A9 87
1762: 02 85 B9 A9 00 85 FB A9 7C
176A: 03 85 FC 20 B7 00 20 37 52
1772: 18 20 B1 00 C9 00 D0 03 DD
177A: 4C 2C 18 C9 40 D0 03 4C 11
1782: 17 18 90 EA C9 43 B0 E6 A6
178A: A2 00 C9 42 D0 02 A2 1A 55
1792: 8E 29 25 20 B1 00 C9 41 5B
179A: 90 66 C9 5B B0 62 38 E9 03
17A2: 40 18 6D 29 25 C9 33 B0 9E
17AA: 57 18 6D 4D 25 A2 41 C9 0D
17B2: 1B 90 05 A2 42 38 E9 1A 3E
17BA: 18 69 40 8D 29 25 8A 20 43
17C2: 37 18 AD 29 25 20 37 18 0B
17CA: 20 B1 00 B0 33 20 4A EC 1C
17D2: 20 36 09 C9 00 D0 29 C0 B2
17DA: 00 F0 25 C0 C9 B0 21 98 E1
17E2: 18 6D 4E 25 A8 A9 00 20 A0
17EA: F2 E2 20 34 ED A2 00 BD 4A
17F2: 00 01 F0 06 20 37 18 E8 D6

```

```

17FA: D0 F5 20 B7 00 4C 76 17 C3
1802: A2 00 BD 80 02 F0 06 9D C0
180A: 00 03 E8 D0 F5 A9 00 9D 19
1812: 00 03 4C 36 18 20 37 18 B7
181A: 20 B1 00 20 37 18 20 B1 D4
1822: 00 20 37 18 20 B1 00 4C D6
182A: 70 17 AC 2A 25 8C 2C 25 69
1832: A9 00 91 FB 60 AC 2A 25 58
183A: C0 78 F0 05 91 FB EE 2A DB
1842: 25 60 AD 45 25 38 ED 41 4E
184A: 25 18 6D 30 25 8D 4B 25 DE
1852: AD 46 25 38 ED 42 25 18 ED
185A: 6D 31 25 8D 4C 25 AD 42 9F
1862: 25 CD 31 25 B0 03 4C 07 42
186A: 19 AD 41 25 CD 30 25 90 17
1872: 4A AD 41 25 8D 47 25 AD 2F
187A: 42 25 8D 48 25 AD 30 25 B0
1882: 8D 49 25 AD 31 25 8D 4A CE
188A: 25 20 A0 16 AD 47 25 CD 6D
1892: 45 25 F0 08 EE 47 25 EE 1B
189A: 49 25 D0 ED AD 48 25 CD 58
18A2: 46 25 F0 14 EE 48 25 EE 70
18AA: 4A 25 AD 41 25 8D 47 25 26
18B2: AD 30 25 8D 49 25 D0 D1 95
18BA: 4C A0 19 AD 45 25 8D 47 58
18C2: 25 AD 4B 25 8D 49 25 AD 36
18CA: 42 25 8D 48 25 AD 31 25 03
18D2: 8D 4A 25 20 A0 16 AD 47 03
18DA: 25 CD 41 25 F0 08 CE 47 18
18E2: 25 CE 49 25 D0 ED AD 48 B6
18EA: 25 CD 46 25 F0 CA EE 48 15
18F2: 25 EE 4A 25 AD 45 25 8D 67
18FA: 47 25 AD 4B 25 ED 49 25 99
1902: D0 D1 4C A0 19 AD 41 25 CB
190A: CD 30 25 90 4A AD 41 25 8D
1912: 8D 47 25 AD 46 25 8D 48 86
191A: 25 AD 30 25 8D 49 25 AD 2C
1922: 4C 25 8D 4A 25 20 A0 16 1B
192A: AD 47 25 CD 45 25 F0 08 2F
1932: EE 47 25 EE 49 25 D0 ED AF
193A: AD 48 25 CD 42 25 F0 14 73
1942: CE 48 25 CE 4A 25 AD 41 03
194A: 25 8D 47 25 AD 30 25 8D B3
1952: 49 25 D0 D1 4C A0 19 AD 6E
195A: 45 25 8D 47 25 AD 4B 25 3A
1962: 8D 49 25 AD 46 25 8D 48 57
196A: 25 AD 4C 25 8D 4A 25 20 76
1972: A0 16 AD 47 25 CD 41 25 AC
197A: F0 08 CE 47 25 CE 49 25 91
1982: D0 ED AD 48 25 CD 42 25 DC

```

198A: F0 14 CE 48 25 CE 4A 25 B6
1992: AD 45 25 8D 47 25 AD 4B DF
199A: 25 8D 49 25 D0 D1 4C 09 AD
19A2: 1C A9 23 A0 99 20 3E 09 8E
19AA: 20 72 10 D0 03 4C 7C 09 E3
19B2: A2 00 A9 08 20 0A 1B F0 3C
19BA: 07 C9 06 F0 03 4C B7 1B 86
19C2: A9 FF 20 95 1B A9 FF 20 C6
19CA: 95 1B A5 6F 20 95 1B A5 6D
19D2: 70 20 95 1B A0 32 B9 F6 E1
19DA: 24 20 95 1B 88 D0 F7 AD B0
19E2: EF 24 85 1B AD F0 24 85 77
19EA: 1C A0 01 B1 1B F0 16 A5 FC
19F2: 1B 20 95 1B A5 1C 20 95 92
19FA: 1B 88 B1 1B 20 95 1B C8 1B
1A02: B1 1B 20 95 1B A5 1B 18 F0
1A0A: 69 02 85 1B A5 1C 69 02 46
1A12: 85 1C A5 1C C5 6C D0 D1 D9
1A1A: A9 FF 20 95 1B A5 6B 85 4C
1A22: 1B A5 6C 85 1C A0 00 B1 48
1A2A: 1B 20 95 1B C8 D0 F8 E6 BA
1A32: 1C A5 1C C5 70 90 F0 F0 56
1A3A: EE 20 52 1B 4C CD 1A A9 61
1A42: 23 A0 9F 20 3E 09 20 72 EE
1A4A: 10 D0 03 4C 7C 09 A2 01 2E
1A52: A9 08 20 0A 1B F0 03 4C F0
1A5A: B7 1B 20 79 1B C9 FF D0 9D
1A62: 60 20 79 1B C9 FF D0 59 F8
1A6A: 20 FA 0A 20 79 1B 85 6F 63
1A72: 20 79 1B 85 70 A0 32 20 5B
1A7A: 79 1B 99 F6 24 88 D0 F7 B1
1A82: 20 79 1B C9 FF F0 18 85 9E
1A8A: 1B 20 79 1B 85 1C 20 79 8B
1A92: 1B A0 00 91 1B 20 79 1B FC
1A9A: A0 01 91 1B 4C 82 1A A5 89
1AA2: 6B 85 1B A5 6C 85 1C A0 FD
1AAA: 00 20 79 1B 91 1B C8 D0 23
1AB2: F8 E6 1C A5 1C C5 70 90 64
1ABA: F0 F0 EE 20 52 1B 4C CD E8
1AC2: 1A 20 52 1B A9 24 A0 DA 02
1ACA: 4C 3E 09 AD C5 B5 D0 08 5F
1AD2: A9 24 A0 BF 20 3E 09 60 61
1ADA: 20 6F 09 A9 00 85 24 85 9A
1AE2: 28 A9 04 85 29 20 80 FE 38
1AEA: AE C5 B5 BD 3F AA AA 8E 03
1AF2: 29 25 BD 71 A9 48 09 80 D4
1AFA: 20 ED FD AE 29 25 E8 68 7D
1B02: 10 ED A9 87 20 F0 FD 60 8A
1B0A: 8D C2 B5 A9 01 8D BB B5 74
1B12: 8D C0 B5 A9 00 8D BD B5 F7


```

1B1A: 8D BE B5 8D BF B5 A9 06 84
1B22: 8D C1 B5 A0 3C A9 A0 99 B3
1B2A: 74 AA 88 D0 FA A0 00 B9 77
1B32: 00 02 F0 08 09 80 99 75 7A
1B3A: AA C8 D0 F3 A9 AA 8D C4 29
1B42: B5 A9 75 8D C3 B5 20 60 DA
1B4A: 1B 20 D6 03 AD C5 B5 60 71
1B52: A9 02 8D BB B5 20 60 1B 55
1B5A: A2 01 20 D6 03 60 A9 00 80
1B62: 8D C7 B5 8D C9 B5 8D CB EC
1B6A: B5 A0 A6 8C CC B5 C8 8C 9C
1B72: CA B5 C8 8C C8 B5 60 8D C8
1B7A: C3 B5 98 48 8A 48 A9 03 63
1B82: 8D BB B5 A9 01 8D BC B5 2D
1B8A: 20 60 1B A2 01 20 D6 03 AF
1B92: 4C AE 1B 8D C3 B5 98 48 45
1B9A: 8A 48 A9 04 8D BB B5 A9 0E
1BA2: 01 8D BC B5 20 60 1B A2 0B
1BAA: 01 20 D6 03 AD C5 B5 F0 55
1BB2: 12 68 68 68 68 AD C5 B5 DA
1BBA: 48 20 52 1B 68 8D C5 B5 D3
1BC2: 4C CD 1A 68 AA 68 A8 AD 52
1BCA: C3 B5 60 20 58 FC 20 84 D9
1BD2: FE A9 06 8D BB B5 8D C1 1E
1BDA: B5 A9 01 8D C0 B5 20 60 CC
1BE2: 1B A2 01 20 D6 03 A9 23 AA
1BEA: 85 FC A9 EC 85 FB 20 54 D7
1BF2: 09 20 12 09 C9 0D D0 F9 A6
1BFA: 20 58 FC 20 22 0B 4C 7C 4B
1C02: 09 AD B3 22 D0 01 60 A9 B7
1C0A: 24 A0 C9 20 3E 09 A5 1D 36
1C12: 8D 30 25 A5 1E 8D 31 25 CA
1C1A: A9 01 85 1D 85 1E AD EF D9
1C22: 24 85 1B AD F0 24 85 1C 4B
1C2A: A0 01 B1 1B F0 35 85 1A 5C
1C32: 88 B1 1B 85 19 B1 19 29 C1
1C3A: 03 C9 02 D0 26 38 20 C7 CD
1C42: 1F A2 00 AC 2C 25 B1 19 EF
1C4A: 8D 2C 25 C8 B1 19 9D 00 B2
1C52: 03 E8 C8 CC 2C 25 D0 F4 B8
1C5A: A9 00 9D 00 03 8E 2C 25 EA
1C62: 20 27 20 A5 1B 18 69 02 E0
1C6A: 85 1B 90 02 E6 1C E6 1E F1
1C72: A5 1E C9 C9 D0 B2 A9 01 80
1C7A: 85 1E E6 1D A5 1D C9 33 14
1C82: D0 A6 AD 30 25 85 1D AD AC
1C8A: 31 25 85 1E 38 20 C7 1F 28

```

```

1C92: 4C 7C 09 20 33 1E 18 20 95
1C9A: C7 1F A9 00 A8 91 1B C8 3E
1CA2: 91 1B 20 03 1C 60 A9 23 77
1CAA: A0 86 20 3E 09 A9 00 2C D7
1CB2: 59 25 30 03 AD 61 C0 0D 98
1CBA: 58 25 30 08 AD B3 22 49 B8
1CC2: FF 8D B3 22 AD B3 22 C9 41
1CCA: 00 F0 06 A9 CE 20 ED FD 6B
1CD2: 60 A9 C6 20 ED FD 20 ED 16
1CDA: FD 60 EE 6A 22 20 B8 09 BA
1CE2: CE 6A 22 AD 00 03 F0 4E 78
1CEA: C9 3D F0 27 AE C4 08 DD 5E
1CF2: C4 08 F0 08 CA D0 F8 A9 63
1CFA: 01 4C 17 1D AD 2C 25 C9 AD
1D02: 25 B0 33 A0 00 A9 03 20 38
1D0A: 81 0C 20 B7 00 D0 E8 A9 46
1D12: 00 F0 02 A9 02 8D 2B 25 25
1D1A: 18 20 C7 1F B0 09 AD B4 0D
1D22: 22 8D 2D 25 4C 32 1D A0 CE
1D2A: 00 B1 19 29 FC 8D 2D 25 24
1D32: 20 27 20 20 03 1C 60 AE 44
1D3A: 36 25 CA CA CA CA BD 00 DB
1D42: 02 C9 45 D0 78 E8 BD 00 88
1D4A: 02 8D 3B 25 E8 BD 00 02 E2
1D52: 38 E9 30 8D 2A 25 E8 BD 77
1D5A: 00 02 38 E9 30 AE 2A 25 70
1D62: F0 06 18 69 0A CA D0 FA 48
1D6A: 8D 29 25 AD 3B 25 C9 2D 64
1D72: F0 4C A2 00 A0 00 BD 00 0D
1D7A: 02 C9 45 F0 08 E8 C9 2E 85
1D82: F0 F4 C8 D0 F1 88 8C 3B 9E
1D8A: 25 AD 29 25 38 ED 3B 25 4F
1D92: 8D 29 25 A2 01 A0 01 BD F6
1D9A: 00 02 E8 C9 2E F0 F8 C9 FF
1DA2: 45 F0 06 99 00 02 C8 D0 80
1DAA: EE A9 30 AE 29 25 99 00 C8
1DB2: 02 C8 CA D0 F9 A9 00 99 96
1DBA: 00 02 8C 36 25 60 CE 29 DB
1DC2: 25 A2 00 A0 00 BD 00 02 3B
1DCA: E8 C9 2E F0 F8 C9 45 F0 2B
1DD2: 06 99 80 02 C8 D0 EE A9 B7
1DDA: 00 99 80 02 A9 2E 8D 00 CC
1DE2: 02 AE 29 25 A9 30 9D 00 8A
1DEA: 02 CA D0 FA A2 00 AC 29 3A
1DF2: 25 C8 BD 80 02 99 00 02 2A
1DFA: F0 04 E8 C8 D0 F4 8C 36 02
1E02: 25 60 20 6F 09 A9 04 85 60
1E0A: 29 A9 00 85 28 85 24 AD EA
1E12: F1 24 38 E5 6F A8 AD F2 22

```

```

1E1A: 24 E5 70 20 21 1E 60 20 54
1E22: F2 E2 20 34 ED A9 01 85 75
1E2A: FC A9 00 85 FB 20 54 09 B9
1E32: 60 A0 01 B1 1B F0 E7 A9 18
1E3A: 00 91 1B 88 91 1B B1 19 3C
1E42: 29 03 C9 02 D0 09 C8 B1 1B
1E4A: 19 A8 B1 19 4C 54 1E C8 BD
1E52: B1 19 85 FB 18 65 19 8D 34
1E5A: 76 1E A5 19 8D 79 1E A5 D3
1E62: 1A 8D 7A 1E 69 00 8D 77 1E
1E6A: 1E A5 70 38 ED 77 1E AA E4
1E72: E8 A0 00 B9 FF FF 99 FF 1A
1E7A: FF C8 D0 F7 EE 77 1E EE 03
1E82: 7A 1E CA D0 EE A5 6F 38 0F
1E8A: E5 FB 85 6F A5 70 E9 00 23
1E92: 85 70 AD EF 24 85 FD AD 43
1E9A: F0 24 85 FE A0 01 B1 FD 63
1EA2: F0 22 38 88 B1 FD E5 19 D9
1EAA: 8D 29 25 C8 B1 FD E5 1A 94
1EB2: 0D 29 25 90 0F 88 B1 FD 69
1EBA: 38 E5 FB 91 FD C8 B1 FD 99
1EC2: E9 00 91 FD C8 F0 03 C8 DE
1ECA: D0 D4 E6 FE C8 A5 FE C5 12
1ED2: 6C D0 C8 60 A9 23 A0 22 36
1EDA: 20 3E 09 20 25 09 C9 59 14
1EE2: D0 03 4C 00 C6 4C 7C 09 3B
1EEA: AD 39 25 85 1D AD 3A 25 82
1EF2: 85 1E 18 20 C7 1F AD 3B CF
1EFA: 25 8D 2B 25 AD 3D 25 8D 1F
1F02: 2D 25 AD 3C 25 8D 2C 25 76
1F0A: 4C 4D 22 48 A5 1D 8D 39 80
1F12: 25 A5 1E 8D 3A 25 AD 2B D5
1F1A: 25 8D 3B 25 AD 2D 25 8D 02
1F22: 3D 25 AD 2C 25 8D 3C 25 BD
1F2A: 68 E9 41 30 BB F0 06 C9 B9
1F32: 02 B0 B5 A9 1A 85 1D 20 30
1F3A: B1 00 E9 40 30 AA F0 A8 49
1F42: C9 1B B0 A4 18 65 1D C9 E6
1F4A: 33 B0 9D 85 1D 20 B1 00 27
1F52: B0 96 20 4A EC 20 36 09 94
1F5A: C9 00 D0 8C C0 00 F0 88 D0
1F62: C0 C9 B0 84 84 1E 38 20 FE
1F6A: C7 1F 90 07 AD 2B 25 C9 05
1F72: 01 D0 03 4C EA 1E A0 02 9D
1F7A: A2 00 B1 19 C9 2A F0 F3 9E
1F82: B1 19 9D 00 02 C8 E8 CC 65
1F8A: 2C 25 D0 F4 A9 00 9D 00 1A
1F92: 02 A5 B8 48 A5 B9 48 A0 1C

```


1F9A: 00 A9 02 20 81 0C 68 85 18
1FA2: B9 68 85 B8 AD 39 25 85 36
1FAA: 1D AD 3A 25 85 1E 18 20 71
1FB2: C7 1F AD 3B 25 8D 2B 25 E0
1FBA: AD 3D 25 8D 2D 25 AD 3C 32
1FC2: 25 8D 2C 25 60 08 A6 1D 5C
1FCA: CA 86 1B A9 C8 85 1C 18 BA
1FD2: A9 00 A2 08 6A 66 1B 90 6E
1FDA: 03 18 65 1C CA 10 F5 85 17
1FE2: 1C A6 1E CA 8A 18 65 1B E3
1FEA: 85 1B A5 1C 69 00 85 1C 9B
1FF2: 06 1B 26 1C A5 1C 6D F0 EA
1FFA: 24 85 1C A0 01 B1 1B D0 10
2002: 03 28 18 60 AA 88 B1 1B CC
200A: 85 19 86 1A 28 90 14 B1 23
2012: 19 29 03 8D 2B 25 B1 19 CC
201A: 29 FC 8D 2D 25 C8 B1 19 7B
2022: 8D 2C 25 38 60 20 33 1E 64
202A: AD 2B 25 C9 02 F0 32 EE 74
2032: 2C 25 EE 2C 25 A0 00 A5 C3
203A: 6F 91 1B C8 A5 70 91 1B B3
2042: 88 AD 2B 25 0D 2D 25 91 E2
204A: 6F C8 AD 2C 25 91 6F C8 04
2052: A2 00 BD 00 03 91 6F C8 A1
205A: E8 CC 2C 25 D0 F4 4C B6 C3
2062: 20 20 F2 20 EE 36 25 EE A4
206A: 36 25 38 AD 36 25 6D 2C 3E
2072: 25 8D 2C 25 AC 36 25 AD B6
207A: 2C 25 91 6F A2 00 C8 BD A7
2082: 00 03 91 6F C8 E8 CC 2C 5C
208A: 25 D0 F4 A0 00 A5 6F 91 41
2092: 1B C8 A5 70 91 1B 88 AD 06
209A: 2B 25 0D 2D 25 91 6F C8 45
20A2: AD 36 25 91 6F C8 A2 02 EA
20AA: BD FE 01 91 6F C8 E8 EC 20
20B2: 36 25 D0 F4 A5 6F 18 6D 49
20BA: 2C 25 90 06 A5 70 C9 A4 F3
20C2: F0 0F A5 6F 18 6D 2C 25 DE
20CA: 85 6F A5 70 69 00 85 70 2C
20D2: 60 A9 00 A8 91 1B C8 91 54
20DA: 1B A9 24 A0 13 20 3E 09 40
20E2: A5 1D 8D F4 24 A5 1E 8D BF
20EA: F5 24 A2 FD 9A 4C 7C 08 6A
20F2: BA 8E 3E 25 A2 00 A0 00 A4
20FA: BD 00 03 C9 28 D0 01 C8 66
2102: C9 29 D0 01 88 9D 00 03 5B
210A: E8 EC 2C 25 D0 EA C0 00 87
2112: F0 03 4C 4D 22 A9 00 48 EB
211A: A9 00 85 B8 A9 03 85 B9 8B
2122: 20 B1 00 90 51 C9 2D F0 E6

```

212A: 4D C9 2B F0 49 C9 2E F0 B8
2132: 45 C9 50 F0 25 C9 28 F0 34
213A: 15 C9 41 F0 0B C9 42 F0 A5
2142: 07 C9 40 F0 0F 4C 4D 22 F7
214A: 20 0D 1F 4C 7B 21 A9 01 3D
2152: 48 4C 22 21 20 05 12 4C A7
215A: 7B 21 20 B1 00 C9 49 F0 6C
2162: 03 4C 4D 22 A9 73 A0 21 82
216A: 20 F9 EA 20 B1 00 4C 7B 3C
2172: 21 82 49 0F DA A1 20 4A E7
217A: EC 20 B7 00 F0 78 A2 02 E2
2182: C9 2B F0 35 E8 C9 2D F0 9F
218A: 30 E8 C9 2A F0 2B E8 C9 CA
2192: 2F F0 26 E8 C9 5E F0 21 C6
219A: C9 29 F0 03 4C 4D 22 68 9E
21A2: F0 14 C9 01 F0 07 48 20 FF
21AA: 19 22 4C A1 21 E6 B8 D0 8C
21B2: 02 E6 B9 4C 7B 21 4C 53 F7
21BA: 12 86 06 68 48 A8 B9 9B E2
21C2: 22 DD 9B 22 90 10 20 19 41
21CA: 22 A6 06 68 48 A8 B9 9B 03
21D2: 22 DD 9B 22 B0 F0 20 72 2F
21DA: EB A5 A2 48 A5 A1 48 A5 3F
21E2: A0 48 A5 9F 48 A5 9E 48 94
21EA: A5 9D 48 A5 06 48 4C 22 D6
21F2: 21 F0 58 4C 69 EA 68 48 E1
21FA: F0 06 20 19 22 4C F8 21 22
2202: 68 20 34 ED A0 00 B9 00 60
220A: 01 99 00 02 F0 03 C8 D0 4B
2212: F5 8C 36 25 4C 39 1D 68 77
221A: 85 FB 68 85 FC 68 85 07 21
2222: 68 85 A5 68 85 A6 68 85 54
222A: A7 68 85 A8 68 85 A9 68 AC
2232: 85 AA 45 A2 85 AB A5 07 E3
223A: 0A A8 A5 FC 48 A5 FB 48 4B
2242: B9 A3 22 48 B9 A2 22 48 F9
224A: A5 9D 60 AE 3E 25 9A A9 25
2252: 07 8D 36 25 A0 00 B9 1B 2A
225A: 23 99 00 02 C8 C0 07 D0 DE
2262: F5 A9 00 99 00 02 60 00 6E
226A: 00 04 04 05 05 06 06 07 D3
2272: 07 04 04 05 05 06 06 07 5F
227A: 07 04 04 05 05 06 06 07 67
2282: 07 00 80 00 80 00 80 00 5F
228A: 80 28 A8 28 A8 28 A8 28 10
2292: A8 50 D0 50 D0 50 D0 50 18
229A: D0 00 01 02 02 03 03 04 AD
22A2: 01 22 01 22 C0 E7 A9 E7 13
22AA: 81 E9 F2 21 96 EE 4E 54 FB
22B2: 46 00 2C 40 40 41 41 41 6E

```

22BA: 42 41 43 41 44 41 45 41 DF
22C2: 46 41 47 41 48 41 49 41 92
22CA: 4A 41 4B 41 4C 41 4D 41 45
22D2: 4E 41 4F 41 50 41 51 41 F7
22DA: 52 41 53 41 54 41 55 41 AA
22E2: 56 41 57 41 58 41 59 41 5D
22EA: 5A 42 41 42 42 42 43 42 1D
22F2: 44 42 45 42 46 42 47 42 C2
22FA: 48 42 49 42 4A 42 4B 42 75
2302: 4C 42 4D 42 4E 42 4F 42 29
230A: 50 42 51 42 52 42 53 42 DB
2312: 54 42 55 42 56 42 57 42 8E
231A: 58 2A 45 52 52 4F 52 2A 83
2322: C5 D8 C9 D4 BA A0 C1 D2 B6
232A: C5 A0 D9 CF D5 A0 D3 D5 62
2332: D2 C5 A0 A8 D9 AF CE A9 C6
233A: BF 00 D3 D0 C5 C5 C4 C3 7A
2342: C1 CC C3 00 D3 D0 C5 C5 48
234A: C4 C3 C1 CC C3 A0 C2 D9 E8
2352: A0 C8 C5 D6 C9 CE A0 CD 9A
235A: C1 D2 D4 C9 CE 00 CE C5 47
2362: D7 BA A0 C1 D2 C5 A0 D9 3C
236A: CF D5 A0 D3 D5 D2 C5 A0 85
2372: A8 D9 AF CE A9 BF 00 D7 8A
237A: C9 C4 D4 C8 BA 00 C7 CF 33
2382: D4 CF BA 00 D2 C5 C3 C1 75
238A: CC C3 D5 CC C1 D4 C9 CF 74
2392: CE A0 C9 D3 A0 CF 00 D3 F6
239A: C1 D6 C5 BA 00 CC CF C1 70
23A2: C4 BA 00 C6 CF D2 CD C1 8D
23AA: D4 BA A0 A0 CC C5 C6 D4 08
23B2: AC A0 C3 C5 CE D4 C5 D2 74
23BA: AC A0 CF D2 A0 D2 C9 C7 52
23C2: C8 D4 A0 CA D5 D3 D4 C9 D4
23CA: C6 D9 BF 00 C6 CF D2 CD CB
23D2: C1 D4 BA A0 A0 A3 A0 CF 35
23DA: C6 A0 C4 C5 C3 C9 CD C1 44
23E2: CC A0 D0 CC C1 C3 C5 D3 1B
23EA: BA 00 8D D0 D2 C5 D3 D3 76
23F2: A0 D2 C5 D4 D5 D2 CE 00 DB
23FA: D0 D2 CF C3 C5 D3 D3 C9 83
2402: CE C7 A0 C4 C1 D4 C1 A0 89
240A: D4 D2 C1 CE D3 C6 C5 D2 AE
2412: 00 CE CF D4 A0 C5 CE CF DE
241A: D5 C7 C8 A0 D2 CF CF CD A5
2422: A0 D4 CF A0 C5 CE D4 C5 CC
242A: D2 A0 C4 C1 D4 C1 00 CD 34
2432: CF D6 C5 A0 C3 D5 D2 D3 C9
243A: CF D2 A0 D4 CF A0 D4 CF FA
2442: D0 A0 CC C5 C6 D4 A0 CF AB


```

244A: C6 A0 CE C5 D7 A0 D0 CF 07
2452: D3 C9 D4 C9 CF CE 00 CD B5
245A: CF D6 C5 A0 C3 D5 D2 D3 F1
2462: CF D2 A0 D4 CF A0 C2 CF FE
246A: D4 D4 CF CD A0 D2 C9 C7 D4
2472: C8 D4 A0 CF C6 A0 C2 CC 70
247A: CF C3 CB 00 D0 D2 C9 CE 49
2482: D4 C9 CE C7 AE AE AE 00 8B
248A: D3 CC CF D4 A0 A3 00 D0 9B
2492: D2 C9 CE D4 A0 D4 CF BA 90
249A: A0 A0 D3 C3 D2 C5 C5 CE 1A
24A2: AC A0 C4 C9 D3 CB A0 CF 7D
24AA: D2 A0 D0 D2 C9 CE D4 C5 C4
24B2: D2 BF 00 C6 C9 CC C5 CE 9C
24BA: C1 CD C5 BA 00 CE CF A0 37
24C2: C5 D2 D2 CF D2 D3 00 D2 B2
24CA: C5 C3 C1 CC C3 D5 CC C1 BC
24D2: D4 C9 CE C7 AE AE AE 00 DB
24DA: CE CF D4 A0 C1 A0 D3 D0 2C
24E2: C5 C5 C4 C3 C1 CC C3 A0 BD
24EA: C6 C9 CC C5 00 00 FF FF FE
24F2: 00 00 FF FF 00 00 FF FF 3B

```

Program 2. Apple SpeedCalc for ProDOS

For mistake-proof program entry, use "AppleMLX" (Appendix C) to type in this program.

START ADDRESS: 2000
END ADDRESS: 3D67

```

2000: 4C A7 3A 00 0A 08 0A 00 1C
2008: A5 AB 33 30 00 14 08 14 E3
2010: 00 8C 32 30 38 33 00 1E 69
2018: 08 1E 00 8C 32 30 38 30 9F
2020: 00 00 00 4C 88 22 20 58 8A
2028: FC AD 61 C0 8D C9 25 A9 12
2030: 00 8D F2 03 A9 09 8D F3 E2
2038: 03 49 A5 8D F4 03 A9 FD DE
2040: 85 39 85 37 A9 1B 85 38 B2
2048: A9 F0 85 36 A9 25 18 69 29
2050: 01 8D 60 25 18 69 4F 85 5D
2058: 6C A9 00 8D 5F 25 8D 61 1E
2060: 25 85 6B 8D D1 22 85 FF FC
2068: 8D C8 25 A9 B9 8D 62 25 CE
2070: A9 09 20 61 09 20 D9 0A 68
2078: A9 23 A0 AE 20 3E 09 20 81
2080: 88 0D 20 25 09 48 20 7C C4
2088: 09 68 AE AC 08 DD AC 08 21
2090: F0 0A CA D0 F8 C9 20 90 F1
2098: E6 4C 37 0C CA 8A 0A AA 46
20A0: A9 08 48 A9 7B 48 BD D3 A7
20A8: 08 48 BD D2 08 48 60 17 1D
20B0: 0E 00 17 06 07 10 03 13 CC
20B8: 0C 18 0A 0B 15 08 02 05 C8
20C0: 21 01 12 04 0D 1B 23 0D 7C
20C8: 31 32 33 34 35 36 37 38 01
20D0: 39 30 2B 2D 2E C4 0A DB 66
20D8: 11 13 10 A8 0C 4E 11 32 E0
20E0: 14 E6 15 9B 19 31 1A 10 13
20E8: 1F DD 10 F6 10 0D 11 37 AF
20F0: 11 CF 1C 16 1D 43 1C 3E FE
20F8: 1E E2 1C 8B 1B C1 15 08 4F
2100: 09 ED 0C 20 58 FC 20 22 DE
2108: 0B 4C 75 08 AD C8 25 49 36
2110: FF 8D C8 25 60 2C 00 C0 95
2118: 10 0B AD 00 C0 8D 10 C0 F7
2120: 29 7F C9 FF 60 A9 00 60 1A
2128: A5 FF F0 07 48 A9 00 85 3A
2130: FF 68 60 20 12 09 F0 FB 2D
2138: 60 20 F2 EB A5 A0 A4 A1 6A
2140: 60 85 FC 84 FB 20 6F 09 44
2148: 20 80 FE A9 00 85 28 85 21
2150: 24 85 25 A9 04 85 29 A0 6E
2158: 00 B1 FB F0 06 20 ED FD 20

```

```

2160: C8 D0 F6 60 A2 32 9D 66 9F
2168: 25 CA D0 FA A9 28 8D 99 5C
2170: 25 60 A0 00 A9 20 99 00 72
2178: 04 C8 C0 28 D0 F6 60 AD 5A
2180: 01 04 C9 10 D0 0A AD 0A 92
2188: 04 C9 02 F0 03 4C 94 09 0A
2190: A9 23 A0 A4 20 3E 09 38 13
2198: 20 02 20 90 03 4C 32 0F 35
21A0: 4C 40 0F 09 80 8D 80 02 C8
21A8: A9 3C 8D 81 02 A2 76 A9 C9
21B0: A0 9D 81 02 CA D0 F8 A0 27
21B8: 01 D0 02 A0 00 B9 80 02 E3
21C0: 8D AC 25 A9 DF 99 80 02 9C
21C8: 20 AB 0A 20 12 09 D0 16 B5
21D0: EE AB 25 10 08 A9 DF 99 5B
21D8: 80 02 4C C5 09 AD AC 25 3F
21E0: 99 80 02 4C C5 09 09 80 F9
21E8: 8D AB 25 AD AC 25 99 80 0A
21F0: 02 AD AB 25 AE 95 0A DD 25
21F8: 95 0A F0 2C CA D0 F8 C9 BE
2200: A0 90 BA 8C AC 25 CE AC 1D
2208: 25 A2 77 BD 80 02 C9 3C 2E
2210: F0 AB CA BD 80 02 9D 81 B5
2218: 02 CA EC AC 25 D0 F4 AD 7C
2220: AB 25 99 80 02 C8 D0 95 29
2228: CA 8A 0A AA BD 9E 0A 48 25
2230: BD 9D 0A 48 60 A0 00 B9 BF
2238: 80 02 C9 3C F0 08 29 7F B3
2240: 99 00 03 C8 D0 F1 A9 00 DF
2248: 99 00 03 8C 9C 25 60 AD 6A
2250: D2 22 F0 20 C0 00 F0 01 8F
2258: 88 4C BA 09 AD D2 22 F0 C9
2260: 13 B9 80 02 C9 3C F0 F1 DF
2268: C8 4C BA 09 AD D2 22 F0 F9
2270: 03 4C BA 09 AD AB 25 29 C0
2278: 7F 85 FF 4C 32 0A C0 00 DD
2280: F0 D7 68 98 AA BD 81 02 1F
2288: 9D 80 02 E8 C9 3C D0 F5 61
2290: A9 A0 9D 80 02 4C BA 09 4D
2298: 07 8D 9B 8A 8B 8B 95 FF 89
22A0: 31 0A 7A 0A 68 0A 68 0A 36
22A8: 4B 0A 58 0A 7A 0A A2 00 02
22B0: BD 80 02 9D 80 04 BD AB 46
22B8: 02 9D 00 05 BD D0 02 9D 88
22C0: 80 05 E8 E0 28 D0 E9 60 6A
22C8: A9 23 A0 C8 20 3E 09 20 77
22D0: 25 09 C9 59 D0 03 20 D9 65
22D8: 0A 4C 7C 09 20 FA 0A A9 FF
22E0: 09 20 61 09 20 22 0B 20 2E
22E8: 88 0D A9 2C 8D 1C 23 A9 79

```



```

22F0: 00 8D 1B 23 A5 6B 85 08 1C
22F8: A5 6C 85 09 60 AD 5F 25 0A
2300: 85 FB AD 60 25 85 FC A0 9D
2308: 00 98 91 FB C8 D0 FB E6 CE
2310: FC A6 FC EC 62 25 D0 F2 29
2318: A9 01 8D 64 25 8D 65 25 BA
2320: 85 1D 85 1E 60 20 28 0B E1
2328: 4C B0 0B A0 05 8C AB 25 03
2330: B9 EB 22 85 28 B9 D3 22 DC
2338: 85 29 A0 00 AE 65 25 A9 9E
2340: 00 8D 99 25 8D 9A 25 F8 89
2348: AD 99 25 18 69 01 8D 99 F5
2350: 25 AD 9A 25 69 00 8D 9A 3B
2358: 25 CA D0 EC D8 A2 00 20 3E
2360: 8D 0B F8 AD 99 25 18 69 25
2368: 01 8D 99 25 AD 9A 25 69 A3
2370: 00 8D 9A 25 D8 EE AB 25 BE
2378: AC AB 25 B9 EB 22 85 28 5B
2380: B9 D3 22 85 29 A0 00 E8 E9
2388: E0 12 D0 D3 20 8D 0B 60 C8
2390: AD 9A 25 18 69 30 91 28 D1
2398: C8 AD 99 25 29 F0 4A 4A 20
23A0: 4A 4A 18 69 30 91 28 C8 19
23A8: AD 99 25 29 0F 18 69 30 3F
23B0: 91 28 60 A0 04 B9 EB 22 E0
23B8: 85 28 B9 D3 22 85 29 A0 5A
23C0: 00 A9 20 91 28 C8 91 28 3E
23C8: C8 91 28 C8 AE 64 25 A9 64
23D0: 00 8D 63 25 BD 66 25 8E 99
23D8: 99 25 4A 69 00 AA CA A9 FE
23E0: 20 91 28 C8 CA D0 FA AD 6A
23E8: 99 25 0A AA BD 1D 23 29 03
23F0: 3F 91 28 C8 BD 1E 23 29 A2
23F8: 3F 91 28 C8 AE 99 25 BD B8
2400: 66 25 4A AA CA CA A9 20 AD
2408: 91 28 C8 CA 10 FA AE 99 4C
2410: 25 BD 66 25 18 6D 63 25 DB
2418: 8D 63 25 E8 BD 66 25 18 1D
2420: 6D 63 25 C9 25 90 AD CA CA
2428: 8E A2 25 A9 20 C0 28 D0 C4
2430: 01 60 91 28 C8 C0 28 D0 30
2438: F9 60 20 A0 09 AD 00 03 A5
2440: F0 3F C9 3D F0 26 AE C4 20
2448: 08 DD C4 08 F0 07 CA D0 2F
2450: F8 A9 01 D0 19 AD 9C 25 8A
2458: C9 25 B0 25 A0 00 A9 03 92
2460: 20 81 0C 20 B7 00 D0 E9 E5
2468: A9 00 F0 02 A9 02 8D 9B CF
2470: 25 AD 1C 23 8D 9D 25 18 B1
2478: 20 02 20 20 62 20 20 3E 69

```

```

2480: 1C 4C 7C 08 85 B9 84 B8 CE
2488: 20 B7 00 4C 4A EC A2 32 11
2490: A9 00 8D A8 25 BD 66 25 FB
2498: 18 6D A8 25 8D A8 25 C9 D2
24A0: 25 B0 03 CA D0 EF E8 E8 B5
24A8: 8E AC 25 60 A9 00 2C C9 7D
24B0: 25 30 03 AD 61 C0 0D C8 C3
24B8: 25 8D C7 25 A0 0D A9 24 F2
24C0: 20 3E 09 20 25 09 C9 4C F8
24C8: F0 0F C9 43 F0 0F C9 52 64
24D0: F0 03 4C 85 0D A2 0C D0 10
24D8: 06 A2 08 D0 02 A2 04 AD 2B
24E0: 1C 23 29 F0 8D AB 25 BA 24
24E8: 0D AB 25 8D AB 25 4C 2F D9
24F0: 0D A9 00 2C C9 25 30 03 33
24F8: AD 61 C0 0D C8 25 8D C7 17
2500: 25 A0 36 A9 24 20 3E 09 8D
2508: 20 76 10 F0 7B A0 00 A9 19
2510: 02 20 81 0C 20 36 09 C9 0A
2518: 00 D0 6D C0 10 B0 69 AD 14
2520: 1C 23 29 0C 8D AB 25 98 25
2528: 0A 0A 0A 0A 0D AB 25 8D CA
2530: AB 25 AD C7 25 10 41 AD 65
2538: AB 25 8D 1C 23 AD 5F 25 C8
2540: 85 1B AD 60 25 85 1C A0 E7
2548: 01 B1 1B F0 11 85 1A 88 4D
2550: B1 1B 85 19 B1 19 29 03 C3
2558: 0D 1C 23 91 19 C8 A5 1B FF
2560: 18 69 02 85 1B A5 1C 69 BA
2568: 00 85 1C A5 1C C5 6C D0 93
2570: D8 38 20 02 20 4C 85 0D A3
2578: 38 20 02 20 90 0A A0 00 17
2580: AD AB 25 0D 9B 25 91 19 AF
2588: 4C 7C 09 A5 1D 8D A0 25 19
2590: A5 1E 8D A1 25 A9 03 8D 64
2598: 63 25 AE 64 25 86 1D AC 24
25A0: 65 25 84 1E 98 18 69 13 64
25A8: 8D 9E 25 BD 66 25 8D A8 6D
25B0: 25 A9 FF EC A0 25 D0 07 09
25B8: CC A1 25 D0 02 A9 3F 8D 46
25C0: A3 25 98 18 69 05 38 ED 78
25C8: 65 25 A8 B9 D3 22 85 29 1B
25D0: B9 EB 22 85 28 38 20 02 F3
25D8: 20 B0 05 A9 A0 4C 67 0E AD
25E0: AD 9B 25 F0 70 C9 02 F0 3C
25E8: 6C AD A8 25 38 ED 9C 25 14
25F0: AA E8 30 32 E8 AD 9D 25 52
25F8: 29 0C C9 08 F0 28 B0 05 23
2600: 8A 4A F0 22 AA 8E A4 25 62
2608: A9 A0 2D A3 25 AC 63 25 F8

```

2610: 91 28 C8 CA D0 FA 8C A5 26
 2618: 25 AD A8 25 38 ED A4 25 B1
 2620: AA A0 02 4C 2E 0E AE A8 9E
 2628: 25 AD 63 25 8D A5 25 A0 1F
 2630: 02 B1 19 8C A4 25 AC A5 8E
 2638: 25 09 80 2D A3 25 91 28 39
 2640: AC A4 25 EE A5 25 CA F0 E7
 2648: 09 C8 CC 9C 25 D0 E2 20 01
 2650: A9 0E 4C 76 0E 20 4E 0F 82
 2658: AE 9C 25 CA CA CA EC A8 78
 2660: 25 B0 03 4C E6 0D A9 2A 79
 2668: 09 80 2D A3 25 AC 63 25 01
 2670: AE A8 25 91 28 C8 CA D0 C6
 2678: FA A4 1E A6 1D C8 CC 9E DD
 2680: 25 F0 05 84 1E 4C A8 0D 05
 2688: AC 65 25 84 1E AD A8 25 8F
 2690: 18 6D 63 25 8D 63 25 E8 30
 2698: 86 1D E0 33 F0 27 BD 66 C4
 26A0: 25 18 6D 63 25 C9 28 B0 BA
 26A8: 1C 4C A8 0D E0 00 F0 14 F8
 26B0: AD 63 25 18 6D A8 25 A8 D3
 26B8: 88 A9 A0 2D A3 25 91 28 97
 26C0: 88 CA D0 FA 60 A9 28 38 FF
 26C8: ED 63 25 8D A8 25 A0 05 82
 26D0: 84 1E B9 D3 22 85 29 B9 8E
 26D8: EB 22 85 28 AC 63 25 AE C2
 26E0: A8 25 A9 A0 91 28 C8 CA 93
 26E8: D0 FA E6 1E A4 1E C0 18 52
 26F0: D0 E0 AD A0 25 85 1D AD C4
 26F8: A1 25 85 1E A0 00 A9 A0 EA
 2700: 99 80 02 C8 C0 78 D0 F8 8A
 2708: 38 20 02 20 90 35 A0 02 59
 2710: A2 00 AD 9B 25 C9 02 D0 44
 2718: 09 AC 9C 25 B1 19 8D 9C A5
 2720: 25 C8 B1 19 09 80 9D 80 01
 2728: 02 E8 C8 CC 9C 25 D0 F2 A5
 2730: A9 3C 9D 80 02 AE 9B 25 45
 2738: BD 18 23 29 3F 8D 27 04 E4
 2740: 4C AB 0A A9 20 8D 27 04 05
 2748: A9 3C 8D 80 02 20 AB 0A 26
 2750: 60 A9 20 8D 00 02 A0 02 61
 2758: B1 19 C9 2A F0 F2 AD 9D ED
 2760: 25 4A 4A 4A 4A 8D A6 25 BC
 2768: A2 FF C9 0F F0 E2 B1 19 C1
 2770: C9 2E D0 09 AE A6 25 F0 25
 2778: 10 E8 8E 00 02 99 FF 01 52
 2780: C8 CC 9C 25 F0 03 CA D0 46
 2788: E5 AD A6 25 F0 1E E0 00 1E
 2790: F0 1A AD 00 02 C9 20 D0 DB
 2798: 0A A9 2E 99 FF 01 C8 AE F9


```

27A0: A6 25 E8 A9 30 99 FF 01 2C
27A8: C8 CA D0 F9 A9 20 8D 00 B0
27B0: 02 CC 9C 25 F0 0C B0 3F 71
27B8: B1 19 C9 2E F0 08 C9 35 B2
27C0: B0 0C C8 4C F4 0F C8 B1 6F
27C8: 19 C9 35 90 2A 88 98 C8 33
27D0: AA CA CA BD 00 02 C9 2E 26
27D8: F0 0B 90 0C C9 39 D0 14 1E
27E0: A9 30 9D 00 02 CA 10 EB 0B
27E8: CA 9D 00 02 E8 A9 31 9D 12
27F0: 00 02 D0 03 FE 00 02 88 8E
27F8: 8C 9C 25 AD 00 02 C9 20 EF
2800: D0 09 A9 01 85 1A A9 FF 28
2808: 85 19 60 A9 01 85 1A A9 04
2810: FE 85 19 EE 9C 25 60 A9 37
2818: 00 2C C9 25 30 03 AD 61 49
2820: C0 0D C8 25 8D C7 25 A9 FE
2828: 23 A0 E1 20 3E 09 20 76 3D
2830: 10 A0 00 A9 02 20 81 0C EA
2838: 20 36 09 C9 00 D0 33 C0 4E
2840: 04 90 2F C0 25 B0 2B A5 90
2848: 1D 8D 64 25 AD C7 25 10 50
2850: 07 98 20 61 09 4C 5B 10 A4
2858: 98 A6 1D 9D 66 25 20 8B AF
2860: 0C A5 1D CD AC 25 90 07 C2
2868: AC AC 25 88 8C 64 25 20 C7
2870: B0 0B 4C 7C 09 A9 01 D0 EE
2878: 02 A9 00 8D A7 25 A0 00 20
2880: A9 1F 20 ED FD A9 88 20 18
2888: ED FD 20 25 09 C9 0D F0 20
2890: 3F C9 0B F0 26 C9 7F F0 4B
2898: 22 C9 20 90 ED AE A7 25 18
28A0: D0 08 C9 30 90 E4 C9 3A 7D
28A8: B0 E0 A6 24 E0 26 F0 DA FC
28B0: 99 00 02 09 80 20 ED FD FC
28B8: C8 D0 C5 C0 00 F0 CB A9 6B
28C0: A0 20 ED FD A9 88 20 ED A4
28C8: FD 20 ED FD 88 4C 7D 10 3E
28D0: A9 A0 20 ED FD A9 00 99 31
28D8: 00 02 8C A6 25 AD 00 02 87
28E0: 60 A5 1E C9 C8 F0 12 E6 40
28E8: 1E AD 65 25 18 69 12 C5 03
28F0: 1E B0 06 EE 65 25 20 28 54
28F8: 0B 60 A5 1E C9 01 F0 10 C1
2900: C6 1E AC 65 25 88 C4 1E 1C
2908: 90 06 CE 65 25 20 28 0B 59
2910: 60 A5 1D C9 32 F0 23 E6 BE
2918: 1D AC A2 25 C4 1D B0 1A E0
2920: EE 64 25 AE 64 25 A9 00 9D
2928: 18 7D 66 25 E8 C9 25 90 4E

```

```

2930: F7 CA CA E4 1D 90 E9 20 F7
2938: B0 0B 60 A5 1D C9 01 F0 0F
2940: 10 C6 1D AC 64 25 88 C4 48
2948: 1D 90 06 CE 64 25 20 B0 A3
2950: 0B 60 A9 23 A0 E8 20 3E CE
2958: 09 20 72 10 A9 01 85 B9 9C
2960: A9 FF 85 B8 20 B1 00 90 1C
2968: 4E 38 E9 41 30 49 F0 06 CF
2970: C9 02 B0 43 A9 1A 8D AB EE
2978: 25 20 B1 00 90 39 38 E9 5F
2980: 40 30 34 F0 32 C9 1B B0 34
2988: 2E 18 6D AB 25 C9 33 B0 C7
2990: 26 8D AB 25 20 B1 00 B0 99
2998: 1E 20 4A EC 20 36 09 C9 CF
29A0: 00 D0 14 C0 00 F0 10 C0 5A
29AB: C9 B0 0C C0 B7 90 0B A9 59
29B0: B6 8D 65 25 4C BA 11 4C 7C
29B8: 7C 09 8C 65 25 84 1E 20 0B
29C0: 8B 0C AD AB 25 CD AC 25 2B
29C8: 90 0A AC AC 25 88 8C 64 0F
29D0: 25 4C D4 11 8D 64 25 85 42
29D8: 1D 20 22 0B 4C 7C 09 AD CA
29E0: 64 25 C5 1D D0 17 AD 65 DC
29E8: 25 C5 1E D0 10 A9 01 8D C6
29F0: 64 25 85 1D 8D 65 25 85 13
29F8: 1E 20 22 0B 60 AD 64 25 FE
2A00: 85 1D AD 65 25 85 1E 60 46
2A08: 20 B1 00 8D BF 25 20 B1 36
2A10: 00 8D C0 25 20 B1 00 8D 87
2A18: C1 25 20 B1 00 C9 28 F0 1E
2A20: 03 4C 88 22 AE 6A 12 AD 2D
2A28: BF 25 DD 6A 12 F0 06 CA 33
2A30: D0 F5 4C 88 22 AD C0 25 EA
2A38: DD 76 12 F0 02 D0 F0 AD 4D
2A40: C1 25 DD 82 12 D0 E8 8E D6
2A48: 99 25 E0 0B B0 0C 8A 48 92
2A50: A9 00 48 4C 5D 21 68 8D 15
2A58: 99 25 20 B1 00 AE 99 25 F4
2A60: CA 8A 0A AA BD 90 12 48 45
2A68: BD 8F 12 48 60 0C 41 41 3D
2A70: 43 45 49 4C 53 53 53 54 88
2A78: 53 41 42 54 4F 58 4E 4F 1C
2A80: 47 49 51 41 55 56 53 4E 02
2A88: 53 50 54 47 4E 4E 52 4E 38
2A90: 4D 45 AE EB 9D F0 E9 EF E5
2A98: 08 EF 22 EC 40 E9 8F EB B4
2AA0: F0 EF 8C EE 39 F0 A9 13 DD
2AAB: 11 14 20 64 13 8E C2 25 52
2AB0: 8C C4 25 20 B7 00 C9 3A AE
2AB8: D0 3F 20 B1 00 20 64 13 C0

```

```

2AC0: 8E C3 25 8C C5 25 20 B7 75
2AC8: 00 C9 29 D0 2C 20 B1 00 07
2AD0: AE C2 25 CA EC C3 25 90 CF
2AD8: 03 4C 88 22 AC C4 25 88 40
2AE0: CC C5 25 90 03 4C 88 22 37
2AE8: E8 C8 A5 1D 8D A9 25 A5 6D
2AF0: 1E 8D AA 25 86 1D 84 1E 2F
2AF8: 60 4C 88 22 18 20 02 20 29
2B00: 90 42 A0 00 B1 19 29 03 8A
2B08: C9 01 F0 38 C8 B1 19 8D F1
2B10: AC 25 A2 00 C8 B1 19 9D 37
2B18: 00 02 E8 C8 CC AC 25 D0 CC
2B20: F4 A5 B8 48 A5 B9 48 A9 44
2B28: 00 9D 00 02 A9 02 A0 00 9C
2B30: 20 81 0C 68 85 B9 68 85 68
2B38: B8 A5 1D CD C3 25 F0 15 7E
2B40: E6 1D 18 60 AD A9 25 85 3E
2B48: 1D AD AA 25 85 1E 18 20 35
2B50: 02 20 4C 88 22 AD C2 25 34
2B58: 85 1D A5 1E CD C5 25 F0 10
2B60: 04 E6 1E 18 60 38 60 A2 FE
2B68: 00 20 B7 00 C9 41 F0 06 F8
2B70: C9 42 D0 D0 A2 1A 8E AB A9
2B78: 25 20 B1 00 C9 41 90 C4 D8
2B80: C9 5B B0 C0 38 E9 40 18 B6
2B88: 6D AB 25 C9 33 B0 B5 8D 17
2B90: AB 25 20 B1 00 B0 AD 20 63
2B98: 4A EC 20 36 09 C9 00 D0 F6
2BA0: A3 C0 00 F0 9F C0 C9 B0 4C
2BA8: 9B AE AB 25 60 A9 01 8D 79
2BB0: 99 25 A9 00 8D 9A 25 20 93
2BB8: A7 12 20 F9 12 B0 47 20 0D
2BC0: 72 EB A5 A2 48 A5 A1 48 8E
2BC8: A5 A0 48 A5 9F 48 A5 9E 85
2BD0: 48 A5 9D 48 EE 99 25 D0 E5
2BD8: 03 EE 9A 25 20 F9 12 08 27
2BE0: 68 8D AB 25 68 85 A5 68 A3
2BE8: 85 A6 68 85 A7 68 85 A8 A3
2BF0: 68 85 A9 68 85 AA 45 A2 9C
2BF8: 85 AB A5 9D 20 C1 E7 AD 11
2C00: AB 25 48 28 90 B9 AD A9 73
2C08: 25 85 1D AD AA 25 85 1E E5
2C10: 18 20 02 20 60 20 AA 13 AA
2C18: A2 06 B5 9C 95 A4 CA D0 69
2C20: F9 AD 9A 25 AC 99 25 20 BC
2C28: F2 E2 A5 AA 45 A2 85 AB 7D
2C30: A5 9D 20 2E 22 60 20 58 D4
2C38: FC A9 01 8D C6 25 A9 00 90
2C40: 2C C9 25 30 03 AD 61 C0 1B
2C48: 0D C8 25 30 03 4C A2 14 A3

```



```

2C50: A9 24 A0 F9 20 3E 09 20 66
2C58: 25 09 C9 53 F0 0B C9 44 7F
2C60: F0 0E C9 50 F0 21 4C AB 43
2C68: 15 A9 03 8D C6 25 D0 35 90
2C70: A9 00 8D C6 25 A0 1D A9 4B
2C78: 25 20 3E 09 20 72 10 20 CE
2C80: E6 1A 90 21 4C 9E 15 A9 A7
2C88: 24 A0 F2 20 3E 09 20 25 F6
2C90: 09 38 E9 30 C9 00 B0 03 6E
2C98: 4C AB 15 C9 08 90 03 4C 16
2CA0: AB 15 8D C6 25 A9 24 A0 EA
2CA8: E6 20 3E 09 20 84 FE AD 93
2CB0: C6 25 F0 14 C9 03 D0 0D 1E
2CB8: AD 05 C3 18 6D 07 C3 C9 FB
2CC0: 50 D0 05 A9 03 20 95 FE 73
2CC8: A5 1D 8D C3 25 8D A0 25 EE
2CD0: A5 1E 8D C5 25 8D A1 25 59
2CD8: A9 01 85 1D 85 1E A9 8D 4E
2CE0: 20 B4 15 A6 1D BD 66 25 55
2CE8: 8D A8 25 AA A9 00 9D 00 0A
2CF0: 03 CA A9 20 9D 00 03 CA 72
2CF8: 10 FA 38 20 02 20 90 58 2B
2D00: AD 9B 25 C9 01 D0 23 AD 98
2D08: A8 25 38 ED 9C 25 AA E8 9D
2D10: 30 14 E8 AD 9D 25 29 0C 5F
2D18: C9 08 F0 0A B0 27 8A 4A 99
2D20: F0 04 AA 4C 42 15 A2 00 B9
2D28: F0 1B 20 4E 0F AE 9C 25 3C
2D30: CA CA CA EC A8 25 90 CF 95
2D38: AE A8 25 A9 2A 9D FF 02 1D
2D40: CA D0 FA F0 13 A0 02 B1 73
2D48: 19 9D 00 03 E8 C8 EC A8 B3
2D50: 25 F0 05 CC 9C 25 D0 EF F1
2D58: A2 00 BD 00 03 F0 08 09 B0
2D60: 80 20 B4 15 E8 D0 F3 A5 03
2D68: 1D CD C3 25 F0 05 E6 1D 16
2D70: 4C E0 14 A5 1E CD C5 25 DE
2D78: F0 0E E6 1E A9 01 85 1D 07
2D80: A9 8D 20 B4 15 4C E0 14 12
2D88: A9 8D 20 B4 15 AD C6 25 7C
2D90: C9 03 D0 03 20 25 09 A9 2C
2D98: 00 20 95 FE AD C6 25 D0 41
2DA0: 03 20 17 1B AD A0 25 85 D8
2DA8: 1D AD A1 25 85 1E 20 58 C0
2DB0: FC 20 22 0B 4C 7C 09 48 35
2DB8: AD C6 25 F0 04 68 4C ED 97
2DC0: FD 68 4C 2B 1B A9 00 2C 1C
2DC8: C9 25 30 03 AD 61 C0 0D 09
2DD0: C8 25 8D AF 25 A9 00 8D E2
2DD8: B0 25 A5 1D 8D B1 25 A5 7E

```

```

2DE0: 1E 8D B2 25 4C 06 16 4C 49
2DE8: 7C 09 A9 00 2C C9 25 30 FB
2DF0: 03 AD 61 C0 0D C8 25 8D D3
2DF8: AF 25 A9 01 8D B0 25 A5 D8
2E00: 1D 8D B1 25 A5 1E 8D B2 4A
2E08: 25 20 41 16 AD A0 25 8D 50
2E10: B5 25 AD A1 25 8D B6 25 52
2E18: 20 4B 16 AE B1 25 CA EC A9
2E20: B5 25 B0 13 AE B2 25 CA 3D
2E28: EC B6 25 B0 0A A9 24 A0 38
2E30: 62 20 3E 09 20 3D 18 AD F1
2E38: B3 25 85 1D AD B4 25 85 4A
2E40: 1E 4C 7C 09 A9 24 A0 C1 BF
2E48: 20 3E 09 4C 52 16 A9 24 8C
2E50: A0 99 20 3E 09 20 88 0D 32
2E58: 20 25 09 AE 86 16 DD 86 E8
2E60: 16 F0 06 CA D0 F8 4C 52 C6
2E68: 16 CA 8A 0A AA A9 16 48 E4
2E70: A9 51 48 BD 8E 16 48 BD F5
2E78: 8D 16 48 60 68 68 A5 1D 7D
2E80: 8D B3 25 A5 1E 8D B4 25 45
2E88: 60 06 00 0B 0A 08 15 0D EE
2E90: DB 11 F6 10 DD 10 37 11 AD
2E98: 0D 11 78 16 AD B9 25 C9 98
2EA0: 33 B0 5B AD BA 25 C9 C9 D0
2EA8: B0 54 AD B7 25 85 1D AD CA
2EB0: B8 25 85 1E 38 20 02 20 AB
2EB8: 90 45 A0 02 AD 9B 25 C9 D2
2EC0: 02 D0 09 AC 9C 25 B1 19 34
2EC8: 8D 9C 25 C8 A2 00 B1 19 D5
2ED0: 9D 00 03 E8 C8 CC 9C 25 C2
2ED8: D0 F4 A9 00 9D 00 03 8E 91
2EE0: 9C 25 20 1E 17 AD B0 25 B0
2EE8: D0 03 20 0C 17 AD B9 25 3B
2EF0: 85 1D AD BA 25 85 1E 18 4C
2EF8: 20 02 20 20 62 20 60 AD ED
2F00: B9 25 85 1D AD BA 25 85 2F
2F08: 1E 18 20 02 20 90 EF 20 E2
2F10: 6E 1E 18 20 02 20 A9 00 16
2F18: A8 91 1B C8 91 1B 4C FB AC
2F20: 16 AD AF 25 30 01 60 AD 31
2F28: 9B 25 C9 02 F0 01 60 AD F0
2F30: B9 25 38 ED B7 25 8D BD C5
2F38: 25 AD BA 25 38 ED B8 25 4E
2F40: 8D BE 25 A2 00 8E 9A 25 78
2F48: BD 00 03 9D 80 02 E8 EC 8A
2F50: 9C 25 D0 F4 A9 00 9D 80 B8
2F58: 02 A9 80 85 B8 A9 02 85 80
2F60: B9 A9 00 85 FB A9 03 85 70
2F68: FC 20 B7 00 20 30 18 20 56

```

2F70: B1 00 C9 00 D0 03 4C 25 31
2F78: 18 C9 40 D0 03 4C 10 18 EB
2F80: 90 EA C9 43 B0 E6 A2 00 B5
2F88: C9 42 D0 02 A2 1A 8E 99 CA
2F90: 25 20 B1 00 C9 41 90 66 9A
2F98: C9 5B B0 62 38 E9 40 18 F0
2FA0: 6D 99 25 C9 33 B0 57 18 80
2FA8: 6D BD 25 A2 41 C9 1B 90 F3
2FB0: 05 A2 42 38 E9 1A 18 69 57
2FB8: 40 8D 99 25 8A 20 30 18 6D
2FC0: AD 99 25 20 30 18 20 B1 D6
2FC8: 00 B0 33 20 4A EC 20 36 38
2FD0: 09 C9 00 D0 29 C0 00 F0 70
2FD8: 25 C0 C9 B0 21 98 18 6D 47
2FE0: BE 25 A8 A9 00 20 F2 E2 E0
2FE8: 20 34 ED A2 00 BD 00 01 44
2FF0: F0 06 20 30 18 E8 D0 F5 4C
2FF8: 20 B7 00 4C 6F 17 A2 00 37
3000: BD 80 02 F0 06 9D 00 03 58
3008: E8 D0 F5 A9 00 9D 00 03 E3
3010: 4C 2F 18 20 30 18 20 B1 3B
3018: 00 20 30 18 20 B1 00 20 EF
3020: 30 18 20 B1 00 4C 69 17 D8
3028: AC 9A 25 8C 9C 25 A9 00 BF
3030: 91 FB 60 AC 9A 25 C0 78 92
3038: F0 05 91 FB EE 9A 25 60 D0
3040: AD B5 25 38 ED B1 25 18 A5
3048: 6D A0 25 8D BB 25 AD B6 89
3050: 25 38 ED B2 25 18 6D A1 40
3058: 25 8D BC 25 AD B2 25 CD E8
3060: A1 25 B0 03 4C 00 19 AD 63
3068: B1 25 CD A0 25 90 4A AD 5C
3070: B1 25 8D B7 25 AD B2 25 8A
3078: 8D B8 25 AD A0 25 8D B9 BB
3080: 25 AD A1 25 8D BA 25 20 27
3088: 99 16 AD B7 25 CD B5 25 5D
3090: F0 08 EE B7 25 EE B9 25 42
3098: D0 ED AD B8 25 CD B6 25 11
30A0: F0 14 EE B8 25 EE BA 25 67
30A8: AD B1 25 8D B7 25 AD A0 18
30B0: 25 8D B9 25 D0 D1 4C 99 90
30B8: 19 AD B5 25 8D B7 25 AD 5D
30C0: B8 25 8D B9 25 AD B2 25 FF
30C8: 8D B8 25 AD A1 25 8D BA 15
30D0: 25 20 99 16 AD B7 25 CD C4
30D8: B1 25 F0 08 CE B7 25 CE 68
30E0: B9 25 D0 ED AD B8 25 CD C8
30E8: B6 25 F0 CA EE B8 25 EE 4C
30F0: BA 25 AD B5 25 8D B7 25 FC
30F8: AD BB 25 8D B9 25 D0 D1 72


```

3100: 4C 99 19 AD B1 25 CD A0 4B
3108: 25 90 4A AD B1 25 8D B7 3A
3110: 25 AD B6 25 8D B8 25 AD E0
3118: A0 25 8D B9 25 AD BC 25 DF
3120: 8D BA 25 20 99 16 AD B7 D6
3128: 25 CD B5 25 F0 08 EE B7 D6
3130: 25 EE B9 25 D0 ED AD B8 BC
3138: 25 CD B2 25 F0 14 CE B8 77
3140: 25 CE BA 25 AD B1 25 8D 9E
3148: B7 25 AD A0 25 8D B9 25 87
3150: D0 D1 4C 99 19 AD B5 25 C2
3158: 8D B7 25 AD BB 25 8D B9 36
3160: 25 AD B6 25 8D B8 25 AD 31
3168: BC 25 8D BA 25 20 99 16 C2
3170: AD B7 25 CD B1 25 F0 08 25
3178: CE B7 25 CE B9 25 D0 ED B3
3180: AD B8 25 CD B2 25 F0 14 89
3188: CE B8 25 CE BA 25 AD B5 8D
3190: 25 8D B7 25 AD BB 25 8D 66
3198: B9 25 D0 D1 4C 44 1C A9 AD
31A0: 24 A0 01 20 3E 09 20 72 20
31A8: 10 D0 03 4C 7C 09 20 E6 9B
31B0: 1A 90 03 4C 40 1B A9 FF 2B
31B8: 20 2B 1B A9 FF 20 2B 1B E5
31C0: A5 08 20 2B 1B A5 09 20 50
31C8: 2B 1B A0 32 B9 66 25 20 90
31D0: 2B 1B 88 D0 F7 AD 5F 25 08
31D8: 85 1B AD 60 25 85 1C A0 98
31E0: 01 B1 1B F0 16 A5 1B 20 40
31E8: 2B 1B A5 1C 20 2B 1B 88 8A
31F0: B1 1B 20 2B 1B C8 B1 1B 24
31F8: 20 2B 1B A5 1B 18 69 02 02
3200: 85 1B A5 1C 69 00 85 1C D6
3208: A5 1C C5 6C D0 D1 A9 FF E6
3210: 20 2B 1B A5 6B 85 1B A5 5A
3218: 6C 85 1C A0 00 B1 1B 20 BE
3220: 2B 1B C8 D0 F8 E6 1C A5 48
3228: 1C C5 09 90 F0 F0 EE 20 7F
3230: 17 1B 4C BB 1A A9 24 A0 8C
3238: 07 20 3E 09 20 72 10 D0 3C
3240: 03 4C 7C 09 20 FA 1A 90 0B
3248: 03 4C 40 1B 20 1E 1B C9 74
3250: FF D0 60 20 1E 1B C9 FF E7
3258: D0 59 20 FA 0A 20 1E 1B 57
3260: 85 08 20 1E 1B 85 09 A0 11
3268: 32 20 1E 1B 99 66 25 88 9C
3270: D0 F7 20 1E 1B C9 FF F0 12
3278: 18 85 1B 20 1E 1B 85 1C 34
3280: 20 1E 1B A0 00 91 1B 20 86
3288: 1E 1B A0 01 91 1B 4C 6F E7

```

```

3290: 1A A5 6B 85 1B A5 6C 85 FE
3298: 1C A0 00 20 1E 1B 91 1B D0
32A0: C8 D0 F8 E6 1C A5 1C C5 A0
32A8: 09 90 F0 F0 EE 20 17 1B 24
32B0: 4C BB 1A 20 17 1B A9 25 0D
32B8: A0 4A 4C 3E 09 60 B0 08 A0
32C0: A9 25 A0 27 20 3E 09 60 36
32C8: 8D 00 02 20 6F 09 A9 00 29
32D0: 85 24 85 28 A9 04 85 29 C5
32D8: 20 80 FE A9 25 A0 31 20 16
32E0: 3E 09 AD 00 02 20 DA FD A0
32E8: 60 20 04 1B 20 00 BF C1 F9
32F0: 62 1B 20 00 BF C0 65 1B 38
32F8: B0 0C 4C FD 1A 20 04 1B 96
3300: 20 00 BF C8 71 1B 60 AC 60
3308: A6 25 B9 00 02 99 01 02 BC
3310: 88 10 F7 AD A6 25 8D 00 7D
3318: 02 60 20 00 BF CC 7F 1B E6
3320: 60 98 48 8A 48 20 00 BF 11
3328: CA 81 1B 4C 38 1B 8D 00 C5
3330: 02 98 48 8A 48 20 00 BF F1
3338: CB 77 1B 90 0F AA 68 68 2B
3340: 68 68 8A 8D 00 02 20 17 7E
3348: 1B 4C C8 1A 68 AA 68 A8 71
3350: AD 00 02 60 02 60 00 BE 24
3358: 01 00 02 04 01 00 B9 00 3B
3360: 02 00 00 01 01 01 00 02 E5
3368: 07 00 02 C3 04 00 00 01 EF
3370: 00 00 00 00 03 00 02 00 F2
3378: BB 00 04 01 00 02 01 00 57
3380: 00 00 01 01 04 01 00 02 3D
3388: 01 00 00 00 4C C5 1A 20 3D
3390: 58 FC 20 84 FE 20 00 BF E6
3398: C5 51 1B B0 EF A0 00 AD 54
33A0: 00 BE 29 0F 8D 00 02 B9 F6
33A8: 01 BE 99 02 02 C8 CC 00 5F
33B0: 02 D0 F4 C8 8C 00 02 A9 89
33B8: 2F 8D 01 02 20 00 BF C6 A1
33C0: 55 1B B0 C8 20 00 BF C8 84
33C8: 71 1B B0 C0 A9 AF 20 ED 0B
33D0: FD 20 00 BF CA 58 1B B0 D8
33D8: B3 A9 B9 85 FC A9 04 85 2F
33E0: FB A0 00 B1 FB D0 08 C8 84
33E8: B1 FB F0 34 4C 09 1C 8D D4
33F0: 99 25 29 0F AA E8 8E A6 40
33F8: 25 C8 B1 FB 09 80 20 ED 92
3400: FD C8 CC A6 25 D0 F3 A9 9B
3408: 8D 20 ED FD A9 27 18 65 5C
3410: FB 85 FB A5 FC 69 00 85 C4
3418: FC C9 BB F0 B4 4C DE 1B A7

```

```

3420: 20 00 BF CC 60 1B A9 24 44
3428: 85 FC A9 54 85 FB 20 54 BD
3430: 09 20 12 09 C9 0D D0 F9 16
3438: 20 58 FC 20 22 0B 4C 7C BA
3440: 09 AD 1B 23 D0 01 60 A9 23
3448: 25 A0 39 20 3E 09 A5 1D 13
3450: 8D A0 25 A5 1E 8D A1 25 36
3458: A9 01 85 1D 85 1E AD 5F B7
3460: 25 85 1B AD 60 25 85 1C B9
3468: A0 01 B1 1B F0 35 85 1A CA
3470: 88 B1 1B 85 19 B1 19 29 30
3478: 03 C9 02 D0 26 38 20 02 76
3480: 20 A2 00 AC 9C 25 B1 19 62
3488: 8D 9C 25 C8 B1 19 9D 00 3D
3490: 03 E8 C8 CC 9C 25 D0 F4 AA
3498: A9 00 9D 00 03 8E 9C 25 3A
34A0: 20 62 20 A5 1B 18 69 02 1E
34A8: 85 1B 90 02 E6 1C E6 1E 60
34B0: A5 1E C9 C9 D0 B2 A9 01 EE
34B8: 85 1E E6 1D A5 1D C9 33 82
34C0: D0 A6 AD A0 25 85 1D AD 22
34C8: A1 25 85 1E 38 20 02 20 44
34D0: 4C 7C 09 20 6E 1E 18 20 DD
34D8: 02 20 A9 00 A8 91 1B C8 0A
34E0: 91 1B 20 3E 1C 60 A9 23 99
34E8: A0 EE 20 3E 09 A9 00 2C 60
34F0: C9 25 30 03 AD 61 C0 0D 3F
34F8: C8 25 30 08 AD 1B 23 49 FE
3500: FF 8D 1B 23 AD 1B 23 C9 4D
3508: 00 F0 06 A9 CE 20 ED FD DA
3510: 60 A9 C6 20 ED FD 20 ED 85
3518: FD 60 EE D2 22 20 B8 09 B0
3520: CE D2 22 AD 00 03 F0 4E 02
3528: C9 3D F0 27 AE C4 08 DD CD
3530: C4 08 F0 08 CA D0 F8 A9 D2
3538: 01 4C 52 1D AD 9C 25 C9 46
3540: 25 B0 33 A0 00 A9 03 20 A6
3548: 81 0C 20 B7 00 D0 E8 A9 B4
3550: 00 F0 02 A9 02 8D 9B 25 74
3558: 18 20 02 20 B0 09 AD 1C 3A
3560: 23 8D 9D 25 4C 6D 1D A0 B8
3568: 00 B1 19 29 FC 8D 9D 25 73
3570: 20 62 20 20 3E 1C 60 AE 5B
3578: A6 25 CA CA CA CA BD 00 82
3580: 02 C9 45 D0 78 E8 BD 00 F6
3588: 02 8D AB 25 E8 BD 00 02 5F
3590: 38 E9 30 8D 9A 25 E8 BD 69
3598: 00 02 38 E9 30 AE 9A 25 BF
35A0: F0 06 18 69 0A CA D0 FA B6
35A8: 8D 99 25 AD AB 25 C9 2D 72

```



```

35B0: F0 4C A2 00 A0 00 BD 00 7B
35B8: 02 C9 45 F0 08 E8 C9 2E F3
35C0: F0 F4 C8 D0 F1 08 8C AB 7D
35C8: 25 AD 99 25 38 ED AB 25 AC
35D0: 8D 99 25 A2 01 A0 01 BD 81
35D8: 00 02 E8 C9 2E F0 F8 C9 6E
35E0: 45 F0 06 99 00 02 C8 D0 EE
35E8: EE A9 30 AE 99 25 99 00 BA
35F0: 02 C8 CA D0 F9 A9 00 99 05
35F8: 00 02 8C A6 25 60 CE 99 C1
3600: 25 A2 00 A0 00 BD 00 02 AA
3608: E8 C9 2E F0 F8 C9 45 F0 9A
3610: 06 99 80 02 C8 D0 EE A9 27
3618: 00 99 80 02 A9 2E 8D 00 3C
3620: 02 AE 99 25 A9 30 9D 00 08
3628: 02 CA D0 FA A2 00 AC 99 1A
3630: 25 C8 BD 80 02 99 00 02 99
3638: F0 04 E8 C8 D0 F4 8C A6 E1
3640: 25 60 20 6F 09 A9 04 85 CE
3648: 29 A9 00 85 28 85 24 AD 59
3650: 61 25 38 E5 08 A8 AD 62 BC
3658: 25 E5 09 20 5C 1E 60 20 30
3660: F2 E2 20 34 ED A9 01 85 E3
3668: FC A9 00 85 FB 20 54 09 28
3670: 60 A0 01 B1 1B F0 E7 A9 86
3678: 00 91 1B 88 91 1B B1 19 AA
3680: 29 03 C9 02 D0 09 C8 B1 89
3688: 19 A8 B1 19 4C 8F 1E C8 19
3690: B1 19 85 FB 18 65 19 8D A2
3698: B1 1E A5 19 8D B4 1E A5 CC
36A0: 1A 8D B5 1E 69 00 8D B2 2F
36A8: 1E A5 09 38 ED B2 1E AA 53
36B0: E8 A0 00 B9 FF FF 99 FF 88
36B8: FF C8 D0 F7 EE B2 1E EE 5E
36C0: B5 1E CA D0 EE A5 08 38 4C
36C8: E5 FB 85 08 A5 09 E9 00 7D
36D0: 85 09 AD 5F 25 85 FD AD D6
36D8: 60 25 85 FE A0 01 B1 FD C9
36E0: F0 22 38 88 B1 FD E5 19 48
36E8: 8D 99 25 C8 B1 FD E5 1A 1F
36F0: 0D 99 25 90 0F 88 B1 FD F3
36F8: 38 E5 FB 91 FD C8 B1 FD 08
3700: E9 00 91 FD C8 F0 03 C8 4E
3708: D0 D4 E6 FE C8 A5 FE C5 81
3710: 6C D0 CB 60 A9 23 A0 8A 0E
3718: 20 3E 09 20 25 09 C9 59 83
3720: D0 03 4C 00 C6 4C 7C 09 AA
3728: AD A9 25 85 1D AD AA 25 EE
3730: 85 1E 18 20 02 20 AD AB 85
3738: 25 8D 9B 25 AD AD 25 8D 5E

```

```

3740: 9D 25 AD AC 25 8D 9C 25 05
3748: 4C 88 22 48 A5 1D 8D A9 2E
3750: 25 A5 1E 8D AA 25 AD 9B 38
3758: 25 8D AB 25 AD 9D 25 8D 40
3760: AD 25 AD 9C 25 8D AC 25 4C
3768: 68 E9 41 30 BB F0 06 C9 28
3770: 02 B0 B5 A9 1A 85 1D 20 9E
3778: B1 00 E9 40 30 AA F0 A8 B7
3780: C9 1B B0 A4 18 65 1D C9 55
3788: 33 B0 9D 85 1D 20 B1 00 95
3790: B0 96 20 4A EC 20 36 09 03
3798: C9 00 D0 8C C0 00 F0 88 3F
37A0: C0 C9 B0 84 84 1E 38 20 6D
37AB: 02 20 90 07 AD 9B 25 C9 92
37B0: 01 D0 03 4C 25 1F A0 02 E1
37B8: A2 00 B1 19 C9 2A F0 F3 0D
37C0: B1 19 9D 00 02 C8 E8 CC D3
37C8: 9C 25 D0 F4 A9 00 9D 00 C0
37D0: 02 A5 B8 48 A5 B9 48 A0 8A
37D8: 00 A9 02 20 81 0C 68 85 86
37E0: B9 68 85 B8 AD A9 25 85 66
37E8: 1D AD AA 25 85 1E 18 20 ED
37F0: 02 20 AD AB 25 8D 9B 25 94
37F8: AD AD 25 8D 9D 25 AD AC B0
3800: 25 8D 9C 25 60 08 A6 1D D9
3808: CA 86 1B A9 C8 85 1C 18 2A
3810: A9 00 A2 08 6A 66 1B 90 DD
3818: 03 18 65 1C CA 10 F5 85 86
3820: 1C A6 1E CA 8A 18 65 1B 53
3828: 85 1B A5 1C 69 00 85 1C 08
3830: 06 1B 26 1C A5 1C 6D 60 C9
3838: 25 85 1C A0 01 B1 1B D0 FF
3840: 03 28 18 60 AA 88 B1 1B 3B
3848: 85 19 86 1A 28 90 14 B1 91
3850: 19 29 03 8D 9B 25 B1 19 BE
3858: 29 FC 8D 9D 25 C8 B1 19 F0
3860: 8D 9C 25 38 60 20 6E 1E 65
3868: AD 9B 25 C9 02 F0 32 EE FE
3870: 9C 25 EE 9C 25 A0 00 A5 71
3878: 08 91 1B C8 A5 09 91 1B D0
3880: 88 AD 9B 25 0D 9D 25 91 21
3888: 08 C8 AD 9C 25 91 08 C8 F6
3890: A2 00 BD 00 03 91 08 C8 41
3898: E8 CC 9C 25 D0 F4 4C F1 7B
38A0: 20 20 2D 21 EE A6 25 EE 2C
38A8: A6 25 38 AD A6 25 6D 9C D8
38B0: 25 8D 9C 25 AC A6 25 AD F4
38B8: 9C 25 91 08 A2 00 C8 BD D7
38C0: 00 03 91 08 C8 E8 CC 9C C4
38C8: 25 D0 F4 A0 00 A5 08 91 E0

```

```

38D0: 1B C8 A5 09 91 1B 88 AD FD
38D8: 9B 25 0D 9D 25 91 08 C8 24
38E0: AD A6 25 91 08 C8 A2 02 3A
38E8: BD FE 01 91 08 C8 E8 EC 53
38F0: A6 25 D0 F4 A5 08 18 6D 52
38F8: 9C 25 90 06 A5 09 C9 B8 11
3900: F0 0F A5 08 18 6D 9C 25 B8
3908: 85 08 A5 09 69 00 85 09 E3
3910: 60 A9 00 A8 91 1B C8 91 C3
3918: 1B A9 24 A0 7B 20 3E 09 F2
3920: A5 1D 8D 64 25 A5 1E 8D 2E
3928: 65 25 A2 FD 9A 4C 7C 08 D1
3930: BA 8E AE 25 A2 00 A0 00 22
3938: BD 00 03 C9 28 D0 01 C8 D5
3940: C9 29 D0 01 88 9D 00 03 C9
3948: E8 EC 9C 25 D0 EA C0 00 04
3950: F0 03 4C 88 22 A9 00 48 0E
3958: A9 00 85 B8 A9 03 85 B9 F9
3960: 20 B1 00 90 51 C9 2D F0 55
3968: 4D C9 2B F0 49 C9 2E F0 27
3970: 45 C9 50 F0 25 C9 28 F0 A2
3978: 15 C9 41 F0 0B C9 42 F0 14
3980: 07 C9 40 F0 0F 4C 88 22 DC
3988: 20 48 1F 4C B6 21 A9 01 54
3990: 48 4C 5D 21 20 05 12 4C 7D
3998: B6 21 20 B1 00 C9 49 F0 78
39A0: 03 4C 88 22 A9 AE A0 21 45
39A8: 20 F9 EA 20 B1 00 4C B6 E5
39B0: 21 82 49 0F DA A1 20 4A 56
39B8: EC 20 B7 00 F0 78 A2 02 51
39C0: C9 2B F0 35 E8 C9 2D F0 0E
39C8: 30 E8 C9 2A F0 2B E8 C9 39
39D0: 2F F0 26 E8 C9 5E F0 21 35
39D8: C9 29 F0 03 4C 88 22 68 F9
39E0: F0 14 C9 01 F0 07 48 20 6E
39E8: 54 22 4C DC 21 E6 B8 D0 4C
39F0: 02 E6 B9 4C B6 21 4C 53 40
39F8: 12 86 06 68 48 A8 B9 03 B8
3A00: 23 DD 03 23 90 10 20 54 69
3A08: 22 A6 06 68 48 A8 B9 03 D9
3A10: 23 DD 03 23 B0 F0 20 72 1C
3A18: EB A5 A2 48 A5 A1 48 A5 AE
3A20: A0 48 A5 9F 48 A5 9E 48 04
3A28: A5 9D 48 A5 06 48 4C 5D 81
3A30: 21 F0 58 4C 69 EA 68 48 51
3A38: F0 06 20 54 22 4C 33 22 BA
3A40: 68 20 34 ED A0 00 B9 00 CE
3A48: 01 99 00 02 F0 03 C8 D0 B9
3A50: F5 8C A6 25 4C 74 1D 68 E0
3A58: 85 FB 68 85 FC 68 85 07 8F

```



```

3A60: 68 85 A5 68 85 A6 68 85 C2
3A68: A7 68 85 A8 68 85 A9 68 1B
3A70: 85 AA 45 A2 85 AB A5 07 52
3A78: 0A A8 A5 FC 48 A5 FB 48 B9
3A80: B9 0B 23 48 B9 0A 23 48 02
3A88: A5 9D 60 AE AE 25 9A A9 17
3A90: 07 8D A6 25 A0 00 B9 83 0F
3A98: 23 99 00 02 C8 C0 07 D0 4D
3AA0: F5 A9 00 99 00 02 60 A9 86
3AA8: 00 85 FB A9 03 85 FD A9 6C
3AB0: 08 85 FC A9 20 85 FE A0 7A
3AB8: 00 B1 FD 91 FB C8 D0 F9 11
3AC0: E6 FC E6 FE A5 FC C9 26 8F
3AC8: D0 EF A9 00 85 F2 20 65 74
3AD0: D6 4C D2 D7 00 00 04 04 A7
3AD8: 05 05 06 06 07 07 04 04 92
3AE0: 05 05 06 06 07 07 04 04 9A
3AE8: 05 05 06 06 07 07 00 80 17
3AF0: 00 80 00 80 00 80 28 A8 88
3AF8: 28 A8 28 A8 28 A8 50 D0 90
3B00: 50 D0 50 D0 50 D0 00 01 B0
3B08: 02 02 03 03 04 3C 22 3C 22
3B10: 22 C0 E7 A9 E7 81 E9 2D A5
3B18: 22 96 EE 4E 54 46 00 2C EF
3B20: 40 40 41 41 41 42 41 43 DB
3B28: 41 44 41 45 41 46 41 47 B9
3B30: 41 48 41 49 41 4A 41 4B 17
3B38: 41 4C 41 4D 41 4E 41 4F 74
3B40: 41 50 41 51 41 52 41 53 D1
3B48: 41 54 41 55 41 56 41 57 2F
3B50: 41 58 41 59 41 5A 42 41 74
3B58: 42 42 42 43 42 44 42 45 E9
3B60: 42 46 42 47 42 48 42 49 47
3B68: 42 4A 42 4B 42 4C 42 4D A4
3B70: 42 4E 42 4F 42 50 42 51 02
3B78: 42 52 42 53 42 54 42 55 5F
3B80: 42 56 42 57 42 58 2A 45 78
3B88: 52 52 4F 52 2A C5 D8 C9 AF
3B90: D4 BA A0 C1 D2 C5 A0 D9 19
3B98: CF D5 A0 D3 D5 D2 C5 A0 E3
3BA0: A8 D9 AF CE A9 BF 00 D3 E4
3BA8: D0 C5 C5 C4 C3 C1 CC C3 80
3BB0: 00 D3 D0 C5 C5 C4 C3 C1 1D
3BB8: CC C3 A0 C2 D9 A0 CB C5 75
3BC0: D6 C9 CE A0 CD C1 D2 D4 E8
3BC8: C9 CE 00 CE C5 D7 BA A0 68
3BD0: C1 D2 C5 A0 D9 CF D5 A0 F9
3BD8: D3 D5 D2 C5 A0 A8 D9 AF 70
3BE0: CE A9 BF 00 D7 C9 C4 D4 65
3BE8: C8 BA 00 C7 CF D4 CF BA 1B

```

```

3BF0: 00 D2 C5 C3 C1 CC C3 D5 AF
3BF8: CC C1 D4 C9 CF CE A0 C9 42
3C00: D3 A0 CF 00 D3 C1 D6 C5 9D
3C08: BA 00 CC CF C1 C4 BA 00 0B
3C10: C6 CF D2 CD C1 D4 BA A0 8E
3C18: A0 CC C5 C6 D4 AC A0 C3 97
3C20: C5 CE D4 C5 D2 AC A0 CF 80
3C28: D2 A0 D2 C9 C7 C8 D4 A0 D4
3C30: CA D5 D3 D4 C9 C6 D9 BF 28
3C38: 00 C6 CF D2 CD C1 D4 BA 63
3C40: A0 A0 A3 A0 CF C6 A0 C4 4F
3C48: C5 C3 C9 CD C1 CC A0 D0 FD
3C50: CC C1 C3 C5 D3 BA 00 8D 8B
3C58: D0 D2 C5 D3 D3 A0 D2 C5 70
3C60: D4 D5 D2 CE 00 D0 D2 CF B8
3C68: C3 C5 D3 D3 C9 CE C7 A0 A5
3C70: C4 C1 D4 C1 A0 D4 D2 C1 32
3C78: CE D3 C6 C5 D2 00 CE CF 86
3C80: D4 A0 C5 CE CF D5 C7 C8 5F
3C88: A0 D2 CF CF CD A0 D4 CF 67
3C90: A0 C5 CE D4 C5 D2 A0 C4 71
3C98: C1 D4 C1 00 CD CF D6 C5 80
3CA0: A0 C3 D5 D2 D3 CF D2 A0 66
3CA8: D4 CF A0 D4 CF D0 A0 CC B0
3CB0: C5 C6 D4 A0 CF C6 A0 CE 0C
3CB8: C5 D7 A0 D0 CF D3 C9 D4 61
3CC0: C9 CF CE 00 CD CF D6 C5 0D
3CC8: A0 C3 D5 D2 D3 CF D2 A0 8E
3CD0: D4 CF A0 C2 CF D4 D4 CF 33
3CD8: CD A0 D2 C9 C7 C8 D4 A0 03
3CE0: CF C6 A0 C2 CC CF C3 CB 2C
3CE8: 00 D0 D2 C9 CE D4 C9 CE B8
3CF0: C7 AE AE AE 00 D3 CC CF 72
3CF8: D4 A0 A3 00 D0 D2 C9 CE AC
3D00: D4 A0 D4 CF BA A0 A0 D3 12
3D08: C3 D2 C5 C5 CE AC A0 C4 5D
3D10: C9 D3 CB A0 CF D2 A0 D0 C3
3D18: D2 C9 CE D4 C5 D2 BF 00 8E
3D20: C6 C9 CC C5 CE C1 CD C5 45
3D28: BA 00 CE CF A0 C5 D2 D2 6B
3D30: CF D2 D3 00 C5 D2 D2 CF B0
3D38: D2 A0 A3 00 D2 C5 C3 C1 AF
3D40: CC C3 D5 CC C1 D4 C9 CE 5D
3D48: C7 AE AE AE 00 CE CF D4 C2
3D50: A0 C1 A0 D3 D0 C5 C5 C4 CA
3D58: C3 C1 CC C3 A0 C6 C9 CC 7B
3D60: C5 00 30 30 3A AD 28 45 E4

```

Memo Diary

Jim Butterfield

Forgetful? Can't seem to remember those important dates? "Memo Diary," a simple, easy-to-use BASIC program, can help you keep track of memorable days, holidays, and personal events. Works with any Apple II series computer using either DOS 3.3 or ProDOS.

"Memo Diary" helps you record and recall birthdays, holidays, appointments, or any other event worth remembering. The program maintains a data file with as many as 100 events whose dates can range from tomorrow to one year in the future. You can even record two different types of dates: temporary, one-time events, such as appointments which have no importance once they've passed, and permanent, recurring events, such as birthdays and anniversaries. By routinely running Memo Diary each time you use your Apple, you'll no longer have to worry about forgetting to mail a birthday card to a relative or finding an anniversary gift for a spouse.

The program always shows the correct day of the week when you enter a date, and you need to enter the year only once—the very first time you run the program. After that (for the next 99 years, anyway), Memo Diary keeps track of the year for you. Each time you run the program, it automatically shows all due and overdue events on the screen or printer, and it erases one-time events from the calendar after they're displayed.

You can enter temporary or recurring new events and erase existing events whenever you wish. You can also examine all events from the current date forward or search the entire calendar for events matching a given starting pattern. Finally, Memo Diary saves your calendar to disk.

Typing the Program

Type in and save a copy of Memo Diary to disk.

The first time you run the program is special. *Do not start the program by entering RUN.* Instead, type RUN 100 and press Return. *If you don't do this, the program will not work correctly.*

When you start the program at line 100, Memo Diary lets you enter the correct month, day, and year without looking for a previous file of events. The year is the most important item

entered at this stage. Type in only the last two digits of the year—86 for 1986, 87 for 1987, and so on. Thereafter, you can start the program with RUN in the usual way.

On the first run, you'll probably want to enter fixed holidays such as New Year's Day and Christmas as well as birthdays and anniversaries. These are permanent events that you won't need to enter year after year. A movable holiday like Thanksgiving should be entered as a one-time event since it falls on a different date each year.

When Memo Diary asks you to enter today's date, you can type in the name of the month (such as OCTOBER) or its number (such as 10). In either case, be careful to enter it correctly. Memo Diary lets you enter any day of the month from 1 to 31, so it won't mind if you tell it the date is February 30. Mistakes like this may confuse the calendar file. For instance, if you use the program on July 4, and the next day give the date as June 5, the computer thinks you've let almost a whole year go by. To warn you of things like this, Memo Diary displays HAPPY NEW YEAR. If you see this message when a new year hasn't arrived, stop the program and start over, entering the correct date.

A Memory Jogger

Except for the very first run, Memo Diary always begins by reporting all due and overdue events ("You just missed your anniversary"). Take careful note of these events, since they'll soon be erased from the calendar (if they're temporary events) or moved ahead to next year (if they're permanent). To help jog your memory, Memo Diary also lets you make a copy of the list of events on your printer.

After disposing of due and overdue events, Memo Diary displays five options: You can see future events, add a new event, cancel an event, search for an event, or quit the program. You'll ordinarily want to look ahead to see what's coming in the next week or two. To do this, choose option 1 (see future events) and supply an appropriate future date when requested. If you enter the current date when looking at future events, Memo Diary assumes you mean the same date *next year* and gives you everything on file.

When you want to make a new entry, select option 2 (add new event). First, Memo Diary asks whether the new event is permanent or one-time. Then it lets you enter the date and de-

tails. Again, the current date is understood as one year from today (it's assumed you don't need to record an event that's happening the same day).

To cancel an event (option 3), you must know its date. When an event is entered, you're shown every item scheduled for that date, each with its own code number. To cancel an event, type in its code number when prompted.

Option 4 (search for event) lets you search for an event based on the first few letters of the entry. You may find many events in the course of a search. For instance, if the calendar file contains the events CLUB MEETING, CLUB CONFERENCE, and CLUB ELECTION, searching for CLUB displays all three events. In this case you would *not* see the entry CANADIAN CLUB, since CLUB is spotted only if it's in the first word of the entry. Thus, if you plan to search for certain keywords (BIRTHDAY, CHURCH, SOFTBALL, or whatever), keep them at the front of each calendar entry.

After you've finished an option, Memo Diary always returns you to the main menu. Sooner or later, you'll be ready to use option 5 (quit). The program knows when it's time to update the calendar file. If you've erased past and overdue events, or added or deleted items, Memo Diary will—with your permission—proceed to update the data file on disk.

The Time Pivot

A program that handles dates can encounter some subtle paradoxes. Does August come *before* April or *after* it? The correct answer is *both*. Memo Diary could resolve this difficulty by adding a year designation to every event, but that complicates the handling of permanent events, which don't belong to a specific year. This isn't a trivial problem: If you schedule a new event for August, the program must decide whether to add the event to the calendar ahead of an existing April event or after it. Without a year designation, how can anyone tell?

The problem is solved by using a *pivot* date, usually the same as the current date. If today is July 4, August does come before April. On the other hand, if today is November 11, April comes before August. Since the calendar always looks one year into the future, everything is kept in order.

However, there's one case in which the pivot date can't be the current date. Each time the program begins, it must

measure the time lapsed since its last use. For example, say you last used the program on August 20, 1986, and next use it on September 4, 1986. On the first run (August 20), Memo Diary uses August 20 as the pivot. That way, an event dated September 1 is seen ahead of another item dated in October.

On the second run (September 4), the September 1 event is reported as past due and is either erased from the calendar (if it's temporary) or moved ahead to September 1 of next year (if it's permanent). Once this is done, the pivot date moves forward to September 4, meaning that a September 1 event now belongs *after* an item dated in October. Don't worry if this sounds confusing: It works out more simply in practice than in theory.

The day of the week is worked out with a simple formula. If you haven't seen it before, here's a hint on how it works. The calendar is modified to make March 1 the first day of the "adjusted year." This way, leap year with its extra February 29 date doesn't break up the sequence of days: The extra leap day just gets pasted onto the year's end. Though the math is a bit convoluted, you may find it interesting to trace the logic of this routine (it starts at line 2150).

Expanding the Calendar

Memo Diary can keep track of a maximum of 100 events. In practice, it's wise to limit the number to 80 or 90 to leave room for permanent events that move automatically from the front to the back of the list. If you need more than 100 events, change the L\$ value in the DIM statement. Line 150 contains the value L\$(100). You can increase the 100 to whatever number you like, but don't get carried away. There's no particular limit to the number of events allowed for a particular date.

Program Notes

Let's take a look at the program's major features. Line 90 prepares Memo Diary to read a file. The variable F is a *Boolean* (logical) variable that's defined as *true* here, to let you read the calendar file on a normal run. When you enter at line 100 on the first run, F is *false* (like every other undefined variable), and no file is read.

DATA statements in lines 110–140 hold the names of the months of the year and days of the week; the names are read into the arrays M\$ and W\$. Line 150 dimensions the L\$ array

for 100 items. Lines 230–250 call for a reading of the calendar file if appropriate. This is done in the subroutine at line 3010. When Memo Diary reads this file, it detects and reports the last date the file was used. Line 260 asks for today's date; the subroutine at line 1670 asks for and accepts the date.

Now it's time to search for due and overdue events. Using the previous date as a pivot, the subroutine at line 1960 scans for all events up to today's date. The program reports these events, erases them, or moves them ahead as needed, and proceeds to the main menu. Line 680 begins a main activity loop: It prompts with the menu, asks for a choice, then goes to the appropriate subroutine. Line 850 lets you see future events. Since the pivot date is now today, the program scans to the requested future date to see how many events fall into the today-to-future-date range.

Line 940 lets you add a new event. After asking *ANNUAL OR ONE-TIME?*, the program requests the event's date and then asks for details. After adding a year designation to the date of one-time events, the new event is inserted into the proper sequence. Line 1210 lets you cancel an event. Memo Diary asks for a date and then lists all events that match that date. At line 1350, the program asks which event to delete. Note that the number you supply must be in the correct range.

Line 1450 begins the search-for-an-event routine. After it receives a search string (P\$), the program looks for a match. When it scans through the calendar, it must look in different places depending on whether the event is one-time or permanent. That's because one-time events carry a year designation, making their dates three characters longer.

A Horrible Mistake?

Line 1570 handles the quit option; the flag F9 registers activity. If you haven't changed any of the data, there's no need to update the calendar file. Before scratching the old file and writing the new one, the program asks whether you're ready. That way, if you make some horrible mistake, you can cancel the file update.

The main loop ends at line 1580 and is followed by several subroutines. The routine starting at line 1590 writes a new calendar file when appropriate, and line 1670 begins the date input routine. The date is formed into a string (D8\$) to allow for easy searches or entry. The subroutine at line 1930 reads

the calendar file. The first item in the file is always the most recent date of use; the remaining data items are events.

The subroutine at line 1960 scans all events to see which have dates between the pivot date (D9\$) and a second date (D8\$). There are three dates involved: event, pivot, and the second date, which makes the comparison a bit messy. Boolean variables keep everything in order. Eventually, the variable F0 indicates the date is in range, and the variable L0 indicates when the last event is found within the date range.

The routine starting at line 4010 displays the information—on the printer if desired. The date is given complete with the day of the week, and events falling on the same day are grouped together. The weekday calculation begins at line 2150. The weekday variable, W, ranges from 0 through 6, so 0 means Sunday. As written, this routine is good for years ranging from 85 (1985) through 84 (2084). If you want to plan more than 99 years in advance, you'll need to modify the routine.

Memo Diary

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
D6 90 F = (1 = 1)
D6 100 GOSUB 2250
84 105 DD$ = CHR$(4):I$ = CHR$(9)
F0 110 DATA JAN,FEB,MAR,APR,MAY,JUN
55 120 DATA JUL,AUG,SEP,OCT,NOV,DEC
16 130 DATA SUNDAY,MONDAY,TUESDAY,WEDNESDAY
BF 140 DATA THURSDAY,FRIDAY,SATURDAY
D4 150 DIM M$(12),W$(6),L$(100)
54 160 FOR J = 1 TO 12
95 170 READ M$(J)
6E 180 NEXT J
56 190 FOR J = 0 TO 6
8D 200 READ W$(J)
61 210 NEXT J
CF 220 PRINT "EVENT CALENDAR"
5A 230 IF F = 0 THEN 260
0F 240 C = 1
CE 250 GOSUB 3010
94 260 PRINT "TODAY'S DATE:"
4A 270 Y8 = Y9
F6 280 GOSUB 1670
0E 290 M8 = M
72 300 D8 = D
AA 310 IF M8 >= M9 THEN 330
```

```
50 320 Y8 = Y9 + 1
B0 330 IF M8 < > M9 OR D8 > = D9 THEN 350
54 340 Y8 = Y9 + 1
4F 350 IF Y8 < = Y9 THEN 370
C1 360 PRINT "HAPPY NEW YEAR"
D9 370 IF F THEN 400
DE 380 PRINT "YEAR";
6A 390 INPUT Y8
B1 400 D9$ = RIGHT$ ( STR$ (100 + M9),2) + "/"
14 410 D9$ = D9$ + RIGHT$ ( STR$ (100 + D9),2)
F0 420 IF F THEN 440
09 430 D9$ = D8$
58 440 F = (1 = 1)
F4 450 GOSUB 1960
B7 460 PRINT "PAST EVENTS: ";
2C 470 IF L0 > = 0 THEN 500
9A 480 PRINT "NONE"
27 490 GOTO 650
4F 500 PRINT L0 + 1
CA 510 GOSUB 4010
AB 520 F9 = - 1
0A 530 FOR J = 0 TO L0
22 540 IF MID$ (L$(J),6,1) = "/" THEN 570
85 550 L$(L9) = L$(J)
BE 560 L9 = L9 + 1
70 570 NEXT J
2E 580 L8 = L0 + 1
63 590 FOR J = L8 TO L9 - 1
97 600 L$(J - L8) = L$(J)
65 610 NEXT J
D2 620 L9 = L9 - L8
B6 630 L8 = 0
2D 640 L = L9
0D 650 F = 0
9C 660 F9 = 0
13 670 D9$ = D8$
4A 680 L = L9 - L8
24 690 IF L < > 0 THEN 710
1C 700 PRINT "NO FUTURE EVENTS"
0C 710 IF L = 0 THEN 730
6F 720 PRINT L;" FUTURE EVENTS"
EC 730 PRINT
11 740 PRINT "1. SEE FUTURE EVENTS"
5A 750 PRINT "2. ADD NEW EVENT"
B1 760 PRINT "3. CANCEL EVENT"
DB 770 PRINT "4. SEARCH FOR EVENT"
6B 780 PRINT "5. QUIT"
FB 790 PRINT
F9 800 PRINT "...YOUR CHOICE (1-5)";
D5 810 INPUT A
```



```

EB 820 PRINT
CA 830 ON A GOTO 850,940,1210,1450,1570
9F 840 GOTO 730
05 850 PRINT "AHEAD TO DATE:"
22 855 FL = 1
F8 860 GOSUB 1670
E3 865 FL = 0
FC 870 GOSUB 1960
AE 875 IF D8$ = D9$ THEN L0 = L9 - 1
3C 880 IF L0 < > - 1 THEN 910
05 890 PRINT "NO EVENTS"
98 900 GOTO 920
CE 910 GOSUB 4010
95 920 PRINT L9 - L0 - 1;" OTHER FUTURE EVENTS"
9E 930 GOTO 730
9A 940 PRINT "ANNUAL OR ONE-TIME (A/O)"
37 950 INPUT P$
E9 960 A = 0
22 970 P$ = LEFT$(P$,1)
E6 980 IF P$ = "O" THEN 1010
10 990 A = 1
96 1000 IF P$ < > "A" THEN 730
7F 1010 GOSUB 1670
F3 1020 Y$ = "/" + RIGHT$(STR$(101 + Y8),2)
90 1030 IF D8$ < = D9$ THEN 1050
F9 1040 Y$ = "/" + RIGHT$(STR$(100 + Y8),2)
B3 1050 IF A < > 1 THEN 1070
09 1060 Y$ = ""
9B 1070 GOSUB 1960
7D 1080 IF L9 - 1 < L0 + 1 THEN 1120
EB 1090 FOR J = L9 - 1 TO L0 + 1 STEP - 1
8C 1100 L$(J + 1) = L$(J)
72 1110 NEXT J
7A 1120 PRINT "DETAIL";
C5 1130 INPUT LL$
31 1140 D8$ = D8$ + Y$
81 1150 D8$ = D8$ + " "
16 1160 L$(L0 + 1) = D8$ + LL$
2B 1170 L9 = L9 + 1
12 1180 L = L9
1D 1190 F9 = - 1
DD 1200 GOTO 680
53 1210 PRINT "CHANGE WHICH DATE:"
87 1220 GOSUB 1670
46 1230 L0 = - 1
5E 1240 FOR J = L8 TO L9 - 1
72 1250 IF D8$ < > LEFT$(L$(J),5) THEN 1300
4F 1260 L1 = J
BD 1270 IF L0 < > - 1 THEN 1290
37 1280 L0 = J

```

```

F6 1290 PRINT J;": ";L$(J)
72 1300 NEXT J
95 1310 IF L0 < > - 1 THEN 1340
95 1320 PRINT "NO EVENTS"
E2 1330 GOTO 730
87 1340 PRINT
28 1350 PRINT " DELETE WHICH EVENT ABOVE";
67 1360 INPUT A
D7 1370 IF A < L0 OR A > L1 THEN 730
5D 1380 FOR J = A TO L9 - 1
45 1390 L$(J) = L$(J + 1)
74 1400 NEXT J
1D 1410 L9 = L9 - 1
07 1420 F9 = - 1
2D 1430 PRINT "...DELETED"
F1 1440 GOTO 680
9E 1450 PRINT "SEARCH FOR";
1A 1460 INPUT P$
7D 1470 P = LEN (P$)
3D 1480 FOR J = 0 TO L9 - 1
49 1490 A = 7
DE 1500 IF MID$(L$(J),6,1) < > "/" THEN 1520
02 1510 A = 10
CF 1520 IF A + P - 1 > LEN (L$(J)) OR P$ < > MID$(L
$(J),A,P) THEN 1540
DA 1530 PRINT L$(J)
86 1540 NEXT J
D7 1550 PRINT "      END OF SEARCH"
F2 1560 GOTO 730
E6 1570 IF F9 < > 0 THEN 1590
EC 1580 END
F4 1590 PRINT "READY TO WRITE NEW EVENTS FILE (Y/N)"
06 1600 INPUT P$
F2 1610 IF LEFT$(P$,1) = "Y" THEN 1630
A3 1620 STOP
EF 1630 D9$ = D9$ + "/"
DB 1640 D9$ = D9$ + RIGHT$(STR$(Y8 + 100),2)
1C 1650 C = 2
74 1660 GOTO 3010
44 1670 M = 0
5F 1680 PRINT "MONTH";
ED 1690 INPUT MM$
D7 1700 M = VAL (MM$)
BD 1710 MM$ = LEFT$(MM$ + "XX",3)
02 1720 IF M = 0 THEN 1760
AC 1730 IF M < 1 OR M > 12 THEN 1670
15 1740 PRINT M$(M)
80 1750 GOTO 1810
66 1760 FOR J = 1 TO 12
90 1770 IF MM$ < > M$(J) THEN 1790

```

```

D0 1780 M = J
9E 1790 NEXT J
A2 1800 IF M < 1 OR M > 12 THEN 1670
CC 1810 PRINT "DAY";
67 1820 INPUT D
24 1830 IF D < 1 OR D > 31 THEN 1670
CA 1840 D8$ = RIGHT$ ( STR$ (100 + M),2) + "/"
35 1850 D8$ = D8$ + RIGHT$ ( STR$ (100 + D),2)
AB 1860 Y = Y8
F7 1865 IF D8$ = D9$ AND FL = 1 THEN 1880
67 1870 IF D8$ > = LEFT$ (D9$,5) THEN 1890
C2 1880 Y = Y8 + 1
8D 1890 GOSUB 2150
87 1900 IF LEN (LL$) < = 0 THEN 1920
D6 1910 PRINT "(:W$(W);)"
EB 1920 RETURN
D9 1930 C = 1
59 1940 GOSUB 3010
F7 1950 RETURN
8C 1960 LL$ = CHR$ (255)
64 1970 L0 = - 1
9B 1980 IF L < > 0 THEN 2000
0B 1990 RETURN
AB 2000 V$ = D8$ + LL$
AB 2010 WW$ = D9$
A0 2030 WW$ = D9$ + LL$
4F 2040 F1 = (WW$ > V$)
5F 2050 FOR J = L8 TO L9 - 1
56 2060 F2 = (L$(J) > WW$)
31 2070 F3 = (V$ > L$(J))
4C 2080 F0 = F2 AND F3
0B 2090 IF F1 = 0 THEN 2110
C2 2100 F0 = F2 OR F3
6B 2110 IF F0 = 0 THEN 2130
1E 2120 L0 = J
7B 2130 NEXT J
E4 2140 RETURN
A0 2150 IF Y > = 85 THEN 2170
ED 2160 Y = Y + 100
F2 2170 M1 = M + 1
6A 2180 M2 = INT (1 / M1 + .7)
46 2190 M3 = Y - M2
A0 2200 M4 = M1 + 12 * M2
0D 2210 N = INT (M4 * 30.6001) + INT (M3 * 365.25) +
D
5E 2220 M6 = INT (N / 7)
26 2230 W = N - 7 * M6
E6 2240 RETURN
51 2250 HOME
EE 2260 RETURN

```



```

71 3000 REM INPUT/OUTPUT ROUTINE
94 3010 F$ = "EVENTS"
76 3020 PRINT DD$;"OPEN ";F$
F8 3030 IF C = 2 THEN 3080
3C 3040 PRINT DD$;"READ ";F$: INPUT LL$:D9$ = LL$: I
    F LEN (LL$) < > 8 THEN PRINT LL$;"?": GOTO 3
    080
FA 3050 M = VAL ( LEFT$ (LL$,2)):D = VAL ( MID$ (LL$
    ,4,2)):Y0 = VAL ( MID$ (LL$,7,2)):M9 = M:D9
    = D:Y9 = Y0:L = 0: PRINT "LAST ACCESS: ";LL$
DA 3060 INPUT L$(L): IF L$(L) < > "EOF" THEN L = L +
    1: GOTO 3060
76 3070 L$(L) = "":L8 = 0:L9 = L: GOTO 3090
42 3080 PRINT DD$;"WRITE ";F$: PRINT D9$: FOR J = 0
    TO L9 - 1: PRINT L$(J): NEXT J: PRINT "EOF"
6A 3090 PRINT DD$;"CLOSE ";F$: IF C = 2 THEN END
D5 3100 RETURN
ED 4000 REM PRINT ROUTINE
F8 4010 PRINT :D$ = "": INPUT "WANT EVENTS ON PRINTE
    R (Y/N) ";P$: IF LEFT$ (P$,1) < > "Y" THEN 4
    030
E4 4020 PRINT DD$;"PR#1": PRINT I$;"80N"
6D 4030 FOR J = L8 TO L0: IF D$ = LEFT$ (L$(J),5) TH
    EN 4060
8A 4040 D$ = LEFT$ (L$(J),5):M = VAL ( LEFT$ (D$,2)
    ):D = VAL ( MID$ (D$,4,2)):Y = Y8: IF D$ < =
    D9$ THEN Y = Y8 + 1
0A 4050 GOSUB 2150: PRINT W$(W);" ";M$(M);" ";D
89 4060 PRINT " "; MID$ (L$(J),6): NEXT J
D6 4070 PRINT : IF LEFT$ (P$,1) = "Y" THEN PRINT DD$
    ;"PR#0"
F4 4080 RETURN

```

The ApWriter

Tim Victor

"ApWriter" lets you write with an Apple II series computer and printer without having to load or learn how to use a large word processing program. It's especially handy for quick tasks like writing letters and memos. Use ApWriter with either DOS 3.3 or ProDOS.

You've got a computer and a printer. How hard could it be to type a couple of lines? You *could* do it in BASIC. First, activate the printer by printing `PR#1`. Don't forget to print `CHR$(4)` first so that DOS will pay attention. Then put each of the lines in `PRINT` statements. Add a bunch of `HTABs` and `VTABs`. It gets to be a lot of work real fast.

What if you use a word processor program instead? You have to load it, which often means rebooting the computer. Then you start typing. If you don't use the program very often, you'll probably have to get out the manual to figure out how to edit your text or even erase your mistakes. Once you've typed in everything, you've got to figure out how to send it to the printer, how to set the margins, how to make it print page numbers, and a lot of other things as well. You go back to the manual. Then you remember that your program can print only from a disk file, which means you'll have to save your text to disk first.

It isn't that word processors are hopelessly difficult or needlessly complex. Each of their features is a response to a particular need that someone has had. A good word processing program provides the power and flexibility that you need for writing term papers, newsletters, and novels. But for quick jobs like writing letters and short memos or making disk labels, a typewriter is quicker and easier.

The "ApWriter" is designed for just these tasks. It lets you type on your computer and print on your printer with the least possible trouble. To make it easier to use, it even works like a typewriter. There are no printing commands, no automatic word-wrapping, and no macros. You just type in a line as you want it printed, hit Return to print it, then move on to the next line.

First of All

ApWriter is written in BASIC, but it uses several short machine language subroutines to speed up its response. There's only one version, but it works for all Apple II computers in either ProDOS or DOS 3.3. Once you've typed in the program, save a copy of it on a disk. Using the "Apple Automatic Proofreader," included in Appendix B, makes it almost impossible for typing errors to creep in.

ApWriter always edits an 80-character line, no matter which Apple II computer you're using or which display mode the computer is in. (If the text screen is 40 characters wide, the edited line occupies two screen lines.) On the bottom screen line, ApWriter displays carets (^) as left and right margin markers. Both markers are on the same line, directly below the corresponding margins, even if the editing line takes up two screen lines.

ApWriter doesn't try to change the computer's display mode. If your Apple II computer can display text on an 80-column screen, *you* choose which mode to use. Whichever mode is active when you begin the program remains active while the program runs.

The different display modes have some minor differences in the appearance of text. On Apple IIc's and enhanced IIe's, certain characters may be displayed incorrectly. When the standard 40-column display mode is active, underlined uppercase letters appear as special graphics characters called *Mousetext*. To display them as they should be seen (in reverse text), you need to turn on the enhanced display. Typing PR#3, then hitting Return, puts the computer in 80-column display mode with the enhanced display activated.

If you'd rather use the 40-column display, press the Escape (Esc) key, followed by the 4 key. The display will be in 40-column mode again, but now uppercase reverse characters will show up properly.

Typing Stuff

When ApWriter is run, the screen clears and two margin markers are displayed on the last screen line. A flashing cursor is located either one or two rows above the left margin marker, depending on the display mode. Each time you press a key, a character is typed on the screen and the cursor moves to the right.

If you make a typing mistake, press either the Delete key or Ctrl-D (press the D key while holding down Control). This erases the character to the left of the cursor. Type the correct character and continue with the rest of the line.

You might not discover a mistake until you've typed several characters after the error. No problem. You can move back and correct it by pressing the left-arrow key to move the cursor left. When the character you want to change is immediately to the left of the cursor, press Delete or Ctrl-D. The character disappears, making the line one character shorter. The rest of the line, including the cursor, moves one space to the left. Now, if you type the correct character, it's inserted to the left of the cursor, making the line one character longer. When you've fixed the mistake, use the right-arrow key to move the cursor back to the right end of the line.

If the line you're editing gets so fouled up that you just want to start over again, press Ctrl-X. The line is erased, the cursor moves back to the left margin, and you can get a fresh start.

ApWriter will beep when there are fewer than five spaces remaining between the cursor and the right margin, just like the bell on a typewriter. Finish the word you're typing, then press the Return key. (If the word you're typing is too long, erase it with the Delete key.) The line is printed on the printer, the screen scrolls up, and the cursor moves to the left margin of a new line.

You can also start a new line by pressing Ctrl-Z. The current line will be printed and a new line started, but the cursor will remain in the same horizontal position instead of returning to the left margin. This can be handy for formatting text or typing columns of figures.

Setting and Clearing Tabs

If you need to move the cursor to a particular horizontal position more than a couple of times, you should set a tab stop at that position. Once a tab has been set, you can quickly move the cursor to that position by pressing either the Tab key or Ctrl-I. The cursor will move to the first tab stop to the right of its original position. If there aren't any tabs to the right, the cursor will move to the right margin instead. To quickly move the cursor to the left margin, press Ctrl-A.

Initially, no tab stops are set, but ApWriter allows you to set as many as 16. They're set and removed by selecting items from the options menu, which is displayed when you press the Esc key. A list of selections appears on the screen, replacing everything except the bottom line of text and the indicators for the margins. A particular menu item can be chosen by pressing the appropriate key.

When the menu is visible, the line of text at the bottom of the screen can't be changed, but the cursor can be moved to the left or right by using the arrow keys. Normally, the cursor's movement is limited by the margins, but in this mode it can be moved all the way to the left and right edges of the screen. Both the Tab key and Ctrl-A are also active in this mode.

To set a tab, move the cursor to the desired column and press the 1 key. To test the new tab, use the left-arrow key to back the cursor up a couple of spaces, then hit the Tab key. The cursor should move over to the new tab stop (unless another stop was already set just to its left).

To remove a tab, press Ctrl-A to move the cursor to the left margin. Then press Tab as many times as needed to move the cursor to the stop you want to remove. Press the 2 key and the stop will be gone. If the cursor isn't at a tab stop when you press the key, nothing will be changed.

Once you've set your tab stops, press 0 to leave the options menu. The menu will be replaced by the text you were working on.

Moving the Margins

On the last line of the screen, a caret (^) is shown directly beneath the left and right margins. ApWriter will let you type only between these margins, but they can be changed from the options menu.

To move the left margin, press the Esc key to display the options menu. Place the cursor at the new margin location and press 3. If the cursor is to the left of the right margin, the left margin will be moved. To move the right margin, the cursor must be to the right of the left margin. When you press 4, the caret marking the right margin is placed below the cursor.

Other Features

The printer can print either single- or double-spaced text. ApWriter prints single-spaced by default, but selection 5 from the options menu lets you flip back and forth between the two settings.

ApWriter can also underline words and sentences. To start underlining, display the options menu, then hit the 6 key. After you press the 0 key to leave the menu, any characters you enter from the cursor position and on will appear on the screen as reverse text (a black character in a white box). Underlining is turned off the same way, by selecting the menu, pressing 6, then pressing 0 to leave the menu. If some of the underlined characters appear garbled on your screen, you may need to select another display mode, as discussed earlier.

On the printer, letters are underlined by printing the character normally, telling the printer to back up one space, then printing an underscore character in the same position. Many printers have a special underline mode to do this automatically, but the codes to turn this function on and off are different on just about every printer.

ApWriter's method works with most printers and interface cards. It's been tested on Apple and Epson dot-matrix and Diablo letter-quality printers. It works with Epson parallel and Apple serial interface cards as well as with the built-in interface on the Apple IIc. Of course, this isn't a complete list. Many other combinations which haven't been tested may also work correctly.

The printer must recognize ASCII code 8 as a backspace, and the interface card must be able to pass this command to the printer. For information about your equipment, check your printer and interface card user's manuals.

Menu selections 7 and 8 are used to advance the paper an entire sheet. Before you start typing, manually advance the paper until the printhead is positioned where you want to print the first line of a page. Then press Esc to select the options menu and hit 7. ApWriter will send a code to your printer, telling it that the current position of the paper is the top of the page. Press 0 to leave the menu and begin typing. When you've filled a page or reached the end of whatever you're typing, go back to the options menu and press 8. The printer should advance the paper to the top of the next page.

ApWriter Commands

Key	Function
Delete (Ctrl-D)	Erases character to left of cursor
Ctrl-X	Erases entire line; moves cursor to left margin
Ctrl-Z	Prints line, but leaves cursor in current position
Tab (Ctrl-I)	Moves cursor to next tab stop
Ctrl-A	Moves cursor to left margin
Esc	Displays options menu

Options Menu

Key	Function
1	Sets tab stop
2	Removes tab stop
3	Sets left margin
4	Sets right margin
5	Single/double space
6	Underlining on/off
7	Sets top of page
8	Advances paper
0	Exits menu

ApWriter

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```

BC 100 PRINT CHR$ (24): POKE 49167,0: GOSUB 1010:D$
    = CHR$ (4):E$ = CHR$ (27):OS = PEEK (190 * 25
    6)
DF 110 IF OS < > 76 THEN O1 = PEEK (43603):O2 = PEEK
    (43604)
CB 120 S$ = "      ": FOR I = 1 TO 4:S$ = S$ + S$: NE
    XT
AB 130 DIM T(16):M = 6:N = 74:LS = 1:UF = 0
46 140 HOME :DM = 80: IF PEEK (64435) < > 6 OR PEEK
    (49183) < 128 THEN DM = 40
A4 150 GOSUB 510: VTAB 1: PRINT
B1 160 V = 23 - (DM = 40)
92 170 B$ = S$:U$ = S$:P = M:W = 0: GOTO 190
81 180 VTAB V: HTAB O: CALL 768, MID$ (B$,O,U)
A1 190 L$ = "":R$ = "": IF P > = N - 5 AND O < N - 5
    THEN PRINT CHR$ (7);

```

```

CF 200 P$ = MID$ (B$,P,1): IF P > 1 THEN L$ = LEFT$
    (B$,P - 1)
33 210 IF P < 80 THEN R$ = RIGHT$ (B$,80 - P)
A8 220 O = P:U = 1: VTAB V: HTAB P: GET C$:C = ASC (
    C$): IF C = 127 THEN 280
2F 230 IF C < 32 THEN 320
E3 240 IF W > = N THEN P$ = ""
80 250 B$ = LEFT$ (L$ + CHR$ (C + 128 * UF) + P$ + R
    $,80): IF P < = W THEN W = W + (W < N):U = U
    + W - P
8A 260 IF P > W AND C < > 160 THEN W = P
12 270 P = P + (P < N): GOTO 180
64 280 IF P = M THEN 180
86 290 IF P < W + 2 THEN O = O - 1:U = W + 2 - P
EA 300 W = W - (P < = W + 1):P = P - 1:B$ = "": IF P
    > 1 THEN B$ = LEFT$ (L$,P - 1)
A6 310 B$ = B$ + P$ + R$ + " ": GOTO 180
3B 320 IF C = 4 THEN 280
EE 330 IF C = 8 THEN P = P - (P > M)
AF 340 IF C = 21 THEN P = P + (P < N)
F5 350 IF C = 13 THEN GOSUB 520: GOSUB 460:P = M
63 360 IF C = 26 THEN GOSUB 460: VTAB V: HTAB 1: CAL
    L 768,B$
CF 370 IF C = 27 THEN A = FRE (0): GOSUB 520: GOSUB
    580: GOSUB 510:P = P + (P < M) * (M - P) - (P
    > N) * (P - N)
72 380 IF C = 9 THEN TP = P: GOSUB 430:P = TP: IF P
    > N THEN P = N
91 390 IF C = 1 THEN P = M
78 400 IF C = 24 THEN B$ = S$:U$ = S$: VTAB V - 1: P
    RINT : CALL 768,B$:P = M
7F 410 IF C = 3 THEN STOP
99 420 GOTO 180
02 430 FOR I = 1 TO 16: IF TP < T(I) THEN TP = T(I):
    I = 17
D5 440 NEXT : IF I = 17 THEN TP = 80
1E 450 RETURN
1E 460 VTAB V: HTAB P: CALL 768,P$: PRINT : GOSUB 10
    00
AF 470 IF W THEN CALL 804, LEFT$ (B$,W)
6A 480 PRINT : IF LS = 2 THEN PRINT
65 490 GOSUB 980
66 500 B$ = S$:U$ = S$:W = 0: VTAB 24: PRINT : IF DM
    = 40 THEN PRINT
59 510 M$ = "^": GOTO 530
48 520 M$ = " "
35 530 POKE 35,25: VTAB 24: IF DM = 40 AND M > 40 TH
    EN HTAB M - 40: GOTO 550
45 540 HTAB M

```

```

47 550 PRINT M$;: IF DM = 40 AND N > 40 THEN HTAB N
    - 40: GOTO 570
C9 560 HTAB N
8F 570 PRINT M$;: POKE 35,24: RETURN
6D 580 CALL 864
A7 590 HP = DM / 2 - 9: HOME : VTAB 6
6E 600 HTAB HP: PRINT "1. TAB SET"
F8 610 HTAB HP: PRINT "2. TAB CLEAR"
D4 620 HTAB HP: PRINT "3. LEFT MARGIN"
52 630 HTAB HP: PRINT "4. RIGHT MARGIN"
35 640 HTAB HP: IF LS = 1 THEN PRINT "5. SET DOUBLE
    SPACING": GOTO 660
2D 650 PRINT "5. SET SINGLE SPACING"
B4 660 HTAB HP: IF UF THEN PRINT "6. END UNDERLINE":
    GOTO 680
62 670 PRINT "6. START UNDERLINE"
4C 680 HTAB HP: PRINT "7. SET TOP OF PAGE"
85 690 HTAB HP: PRINT "8. NEXT PAGE"
BD 700 HTAB HP: PRINT "0. QUIT"
78 710 VTAB V: CALL 768,B$
EC 720 GOSUB 510:MP = P
AE 730 VTAB V: HTAB MP: GET A$: IF A$ = CHR$ (8) THE
    N MP = MP - (MP > 1)
16 740 IF A$ = CHR$ (21) THEN MP = MP + (MP < 80)
72 750 IF A$ = CHR$ (1) THEN MP = M
89 760 IF A$ = CHR$ (9) THEN TP = MP: GOSUB 430:MP =
    TP
6E 770 IF A$ = "0" THEN CALL 889: RETURN
6A 780 IF A$ < > "1" THEN 840
4E 790 I = 1
48 800 IF T(I) = MP THEN 840
35 810 IF T(I) < MP AND T(I) < > 0 THEN I = I + 1: I
    F I < 16 THEN 800
0A 820 IF I < 16 THEN FOR J = 15 TO I STEP - 1:T(J +
    1) = T(J): NEXT
06 830 T(I) = MP
62 840 IF A$ < > "2" THEN 910
47 850 I = 1
52 860 IF T(1) = MP THEN 890
FA 870 I = I + 1: IF I < 17 THEN 860
A6 880 GOTO 910
C2 890 IF I < 16 THEN FOR I = I TO 15:T(I) = T(I + 1
    ): NEXT
E6 900 T(16) = 0
35 910 IF A$ = "3" AND MP < N THEN GOSUB 520:M = MP:
    GOSUB 510
B9 920 IF A$ = "4" AND MP > M THEN GOSUB 520:N = MP:
    GOSUB 510
FA 930 IF A$ = "5" THEN LS = 3 - LS: GOTO 590

```



```

50 940 IF A$ = "6" THEN UF = 1 - UF: GOTO 590
46 950 IF A$ = "7" THEN GOSUB 1000: PRINT E$; CHR$ (
    118);: GOSUB 980
BC 960 IF A$ = "8" THEN GOSUB 1000: PRINT CHR$ (12);
    : GOSUB 980
A6 970 GOTO 730
3D 980 IF OS < > 76 THEN POKE 43603,01: POKE 43604,0
    2: RETURN
6B 990 PRINT D$;"PR#0": RETURN
F5 1000 VTAB V: PRINT : PRINT D$;"PR#1": PRINT CHR$
    (9);"255N";: RETURN
9F 1010 FOR I = 768 TO I + 183: READ A: POKE I,A: NE
    XT : RETURN
35 1020 DATA 32,177,0,240,23,32
23 1030 DATA 123,221,32,108,221,32
0B 1040 DATA 0,230,170,160,0,177
1B 1050 DATA 94,73,128,32,237,253
CF 1060 DATA 200,202,208,245,96,0
34 1070 DATA 0,0,0,0,0,0
EE 1080 DATA 32,177,0,240,39,32
3B 1090 DATA 123,221,32,108,221,32
FC 1100 DATA 0,230,170,160,0,177
11 1110 DATA 94,8,9,128,32,237
EE 1120 DATA 253,40,16,12,72,169
CB 1130 DATA 136,32,237,253,169,223
5F 1140 DATA 32,237,253,104,200,202
72 1150 DATA 208,229,96,0,0,0
32 1160 DATA 0,0,0,0,0,0
36 1170 DATA 0,0,0,0,0,0
7F 1180 DATA 32,160,3,177,157,145
0E 1190 DATA 159,141,85,192,177,157
50 1200 DATA 141,84,192,145,161,200
0B 1210 DATA 208,239,32,146,3,208
D1 1220 DATA 234,32,160,3,177,159
17 1230 DATA 145,157,177,161,141,85
FE 1240 DATA 192,145,157,141,84,192
7E 1250 DATA 200,208,239,32,146,3
BC 1260 DATA 208,234,230,158,230,160
40 1270 DATA 230,162,206,184,3,208
64 1280 DATA 2,104,104,96,169,4
E3 1290 DATA 133,158,141,184,3,169
45 1300 DATA 64,133,160,169,68,133
DA 1310 DATA 162,160,0,132,157,132
BF 1320 DATA 159,132,161,96

```

Your Personal Ledger

Alan H. Stein

This powerful, but simple-to-use financial application helps you track your expenses, income, assets, liabilities, even your taxable expenditures. You can quickly enter transactions, sort them, and print comprehensive reports. For all Apple II series computers using either DOS 3.3 or ProDOS.

"Your Personal Ledger" is exceptionally easy to use, yet surprisingly powerful. Each financial transaction can be coded in two different ways, each with a choice of 26 different user-generated codes. Naturally, if a mistake is made, you can edit or even delete transactions. Reports can be printed with records chosen by date and/or code. It's a simple matter to get a list of all expenses for a given period of time along with a total for those expenses.

Your Personal Ledger is completely written in Applesoft BASIC and is compatible with both ProDOS and DOS 3.3.

Starting Your Personal Ledger

Once you've entered Your Personal Ledger (make sure to use the "Apple Automatic Proofreader" error-checking program, found in Appendix B, to help you avoid typing mistakes) and saved it to disk, you're ready to begin.

The easiest way to see how to use Your Personal Ledger is simply to go through a sample tutorial session. Let's begin.

Place the disk containing Your Personal Ledger in the disk drive (drive 1 if you have a dual-drive system), turn the computer on, and run the program by typing `RUN LEDGER` (or whatever other name you may have given the file). The first screen you'll see shows the following display:

```
PRESS TO
-----
1      USE EXISTING LEDGER
2      SET UP NEW LEDGER
```

Since you've never used Your Personal Ledger before, press 2. You'll be instructed to put a formatted disk in drive 1

and press Return. (If you'd previously used Your Personal Ledger, you would have pressed 1).

The next thing you'll see is the main menu:

```
PRESS TO
-----
1      EDIT TRANSACTION CODES
2      ENTER TRANSACTION
3      FIND/CHANGE TRANSACTION
4      SORT TRANSACTIONS
5      PRINT REPORT

<ESC>  SAVE CHANGES/EXIT
```

Since you've not used Your Personal Ledger before, it's best to first set up some transaction codes. Press 1.

Because there are two different sets of transaction codes available, you're asked which you want to edit. Start with code 1, and you'll see the following display:

```
EDIT TYPE 1 TRANSACTION CODE
CODE  DESCRIPTION
A
PRESS  <--OR-->  FOR OTHER CODES
        <C>      TO CHANGE
        <ESC>     FOR MENU
```

Each transaction code type has 26 codes, A-Z, each of which has a description. Since you've not assigned any descriptions yet, the DESCRIPTION column is blank. Let's assign some type 1 codes. Almost everyone will find categories for *income*, *expenses*, *assets*, *liabilities*, and *other* useful. These descriptions can be changed anytime, but it's a good idea to do some advance planning.

To assign the description INCOME to code A, press C (for change) and then type INCOME. Of course, press Return after entering INCOME to let your Apple know when you're done. Your Personal Ledger generally doesn't require a press of the Return key for single-key responses, but it does need it when the response can be of an unknown length.

After typing in the description, you'll see it displayed where only a blank space appeared before. Now, assign the description EXPENSE to code B. That's almost as simple—just press the right-arrow key, and B appears beneath CODE. You can then type in the description EXPENSE. In a similar way, assign ASSET to code C and LIABILITY to code D. But what if

you want to assign OTHER to code Z? You can display code Z either by pressing the right-arrow key several times, thus going through the alphabet, or by pressing the left-arrow key a few times and backing up (using the left-arrow key shows Z immediately after A). When you're finished, press Esc to return to the main menu.

So far, you've just set up a few type 1 codes. Now for type 2 codes (you can think of type 2 codes as subcodes of a sort). Again from the main menu, choose EDIT TRANSACTION CODES, this time picking type 2 codes. They're entered the same way as type 1 codes. To start with, enter descriptions of *salary, interest income, rent, food, utilities, entertainment, and, naturally, other.*

Transactions

You're now ready to enter some transactions—press 2 from the main menu. You'll see prompts on the screen for a date, check number, codes, source/payee, description, and amount.

The date must be entered in the form *mm/dd/yy*. The slashes are supplied automatically, and the Return key does *not* have to be pressed. However, all single-digit numbers must be entered with leading zeros. For instance, June 5, 1986, would be entered by typing *060586*, and displayed as *06/05/86*.

The check number can either be entered or omitted by pressing Return. You can also enter nonnumerical information here—for example, a MasterCard purchase might be entered with MC in the check number column. Numbers longer than five digits are truncated, leaving only the *last* five digits. If you're entering text here, the same restriction applies.

The codes are entered in a manner similar to the way they were edited—press the arrow keys until the desired code comes into view, then press Return.

Source/Payee represents either the source of income or the person or company to whom a payment is made. For example, if you received a check from *COMPUTE!*, you could type *COMPUTE*, while if you paid for your groceries by check, you'd enter the name of the market. Your Personal Ledger allows 20 characters for this entry.

Another 20 characters are allowed for the description, which might be *March groceries* or *sub to COMPUTE!*

Finally, the amount would be entered, using only numbers and a decimal point, not the dollar sign.

After entering each transaction, you're given the choice of pressing the Esc key to abort the transaction (and thus returning to the menu) or pressing Return to accept the transaction. If you accept the transaction, you're taken automatically to the FIND/CHANGE menu, which can also be reached directly from the main menu (option 3).

FIND/DISPLAY TRANSACTION

DATE:	09/12/86
CHECK NO:	1921
CODE 1:	LIABILITY
CODE 2:	SALARY
SOURCE/PAYEE:	FIRST CITIZENS BANK
DESCRIPTION:	MARCH PAYMENT
AMOUNT:	195.70

<--	-->
<C>HANGE	<J>UMP
<PRINT>	<D>ELETE
<E>NTER	<ESC> MENU

You can easily view other transactions (assuming you've entered more than just one) by pressing either arrow key or by pressing the J key. Pressing an arrow key moves forward or backward by one transaction, while pressing J gives you the opportunity to move quickly through the file by specifying a number of transactions to move. Entering a positive number moves forward, while entering a negative number moves backward in the file. (If you want to move forward, you *don't* need to put a plus sign, +, before the number of transactions to jump.)

Pressing P prints the transaction information if you have a printer connected properly. Pressing D lets you delete the transaction. By this time, you probably haven't entered any transactions you want to delete, but keep this option in mind for later—you'll certainly find a use for it.

Pressing C gives you a chance to change any of the information in a transaction. You're led through each transaction and

given the opportunity either to reenter the information—in the same manner it was originally entered—or to leave it as is.

Hitting E lets you enter another transaction without returning to the main menu. Press Esc to get back to the main menu.

Now Sorting, Now Reporting

The fourth choice from the main menu, SORT TRANSACTIONS, rearranges the transactions in chronological order. It also compresses the information in memory. Your Personal Ledger is set up for a maximum of 400 transactions. However, transactions deleted in a session count toward that maximum until the next session, *unless* the transactions are sorted. Thus, if you're unable to enter more transactions, you may be able to make room simply by sorting.

Option 5 is PRINT REPORT, the facility which enables you to get a hardcopy of the data you've put in your file.

When you print a report, the first thing you're asked is whether you want to use the default format. This format prints all fields at a predetermined width. If you decide to devise your own format, you're shown the default field widths for each field. You can accept each default simply by pressing Return, or you can change it by pressing C and entering the new width. If you want to omit a field from the report, define its width as zero. (Any custom format becomes the default format for the rest of the session.)

Let's say that you want to print a listing of all your utility bills—you won't need the second code type printed on this report. Define your own format by accepting all defaults except for the type 2 code, which you should change to a width of zero.

After determining the format, you must determine which records are to be included. The first display you'll see looks like this:

```
SELECTION
PRESS <RETURN> TO ACCEPT ALL DATES
OR
PRESS <C> TO CHOOSE PARTICULAR DATES
```

If you press Return, the date of each transaction will not be a factor in determining whether to include it. If you press C, then you'll be asked for a time span—that is, an earliest date and a latest date. Provide the month and year of these

dates in the form of two digits for the month and two digits for the year (0986 could be an example).

Next, you'll need to specify which codes to accept for each type (1 and 2) code. In each case, you can choose to include records with any code by simply pressing Return, or choose records which have a particular code by specifying it.

Since you want a listing of utility bills, you'll press Return for each selection criterion, except for the type 2 code, which you should specify as UTILITY.

Once the selection process is complete, Your Personal Ledger prints the report, complete with a total at the bottom, and goes back to the main menu.

Reporting with Your Personal Ledger

PERSONAL LEDGER REPORT

DATE	CHECK CODE 1	CODE 2	SOURCE/PAYEE	DESCRIPTION	AMOUNT
07/21/86	235 EXPENSE	FOOD	HARRY'S GROCERY	WEEKLY MARKETIN	121.73
07/22/86	236 EXPENSE	RENT	WATERSIDE CONDO	AUGUST RENT	435.00
07/23/86	237 EXPENSE	ENTERTAI	BOB'S GRILL	DINNER W/ EMILY	23.67
07/25/86	238 EXPENSE	UTILITIE	DUKE POWER	JULY ELECTRIC	69.96

TOTAL: \$650.36

Saving Grace

When you're finished, press the Esc key when the main menu is displayed. This selects the SAVE CHANGES/EXIT option. This is how you exit Your Personal Ledger or how you periodically back up your data file during a long session. (You have one last chance to go back to the menu without saving the changes by pressing Esc again.) Pressing Return saves the changes to disk. At this point, you can reenter Your Personal Ledger by hitting the Esc key or end the session entirely by again pressing the Return key.

More Personal

Your Personal Ledger is so simple, yet so powerful, that there's probably little you might want to change. It's set up for a maximum of 400 transactions in order to avoid running out of memory. This is a conservative figure chosen to avoid either having to carefully calculate how much RAM was available for DATA or having to put in sophisticated error-detection code to avoid termination of the program if it ran out of memory. That maximum can easily be changed (change the value assigned to

MR in line 190) and is an option if your Apple has additional memory and an enhanced version of Applesoft BASIC able to handle that memory.

If your Apple II computer doesn't have this capability, you can still record more than 400 transactions, though this requires additional disks. For instance, you may have to keep two (or more) data disks available. Simply label each disk with the appropriate time frame, say *January-June 1986*. If you have a multitude of transactions to record, you may even have to go to a one-disk-per-month system. (You'll have to use additional disks since Your Personal Ledger always saves its data to a file called LEDGER.DATA.)

In addition, you may be interested in a graphic depiction of your financial status. The system used to store the data on disk is simple enough so that it would not be difficult to extract the data in a form usable by a stand-alone graphics program (such as *Visiplot/Visitrend*) or a simple program you might write.

Your Personal Ledger is an easy-to-use, yet flexible system for keeping track of your financial situation. Indeed, if you'd spent the time it took to read this article using Your Personal Ledger, you'd already have figured out how much you went over budget last month.

How Your Personal Ledger Works

Your Personal Ledger stores all transactions in an array called RD\$. Each element in the array corresponds to a transaction. The first six characters represent the date, stored in the form *yyymmdd*. The next five characters contain the check number. The following two characters contain the type 1 and type 2 codes (a memory-saving technique). Twenty characters each are set aside for the source/payee and description, with the last nine characters representing the amount of the transaction.

The actual descriptions of the codes are stored in an array CODE\$.

When the program executes, these code descriptions along with all the transaction data are read from the sequential disk file LEDGER.DATA. The actual number of transactions is also stored in that file and is assigned to the variable NR. All this is performed in lines 380-510. (When setting up a new file, this step is skipped. All elements of CODE\$ and RD\$ are automatically initialized to null strings and NT is initialized to

zero.) A pointer to the highest element of the RD\$ array, TR, is initialized to the same value as NR.

Lines 750–1060 perform the assignment of descriptions to the transaction codes by simply setting elements of the array CODE\$ to the description entered.

Lines 1070–1240 control the entry of new transactions. Separate subroutines are used for each field. When the entry is complete, the number of records is incremented as well as the pointer to the top record in the array RD\$. Lines 4150–4180 combine the different fields into a single character string RD\$, which is then stored in the array RD\$.

A variable RN is used to point to the last transaction accessed. When the option FIND/CHANGE TRANSACTION is chosen, the transaction pointed to by RN is retrieved and separated into individual fields by the subroutine on lines 4190–4270. If another transaction is desired, the value of RN is increased or decreased and the process is repeated. If the transaction is to be changed, the same subroutines used for entry are used to change the fields.

When a transaction is deleted, the corresponding entry in the array RD\$ is set to the null character string and NR is decremented. This is performed by lines 1810–1920.

Sorting of transactions is performed in lines 2030–2210. A fairly straightforward procedure similar to a bubble sort is used. Each element of the array RD\$ is processed in turn. The element being processed is simply compared to each of those already processed and swapped until it has reached its proper place relative to the elements already processed. Deleted entries are easily recognized since they are stored as null strings and are skipped over. When the process is complete, the pointer TR is reset to the same value as NR.

Although this sort process is relatively primitive, it's fairly effective in this situation since most of the array will already be in order.

The printing of a report (lines 2220–3190) is straightforward. Each transaction is printed, with the length of each field determined by the value of a variable set either when the program started or when the user decided to determine a custom format. If minimum or maximum dates were used, any record not falling within that time span is ignored. Similarly, if either code type was specified, any record with a different code is ignored. The total amounts of all the transactions printed are

stored in a variable A. When all the transactions have been printed, the value of A is then printed as well after some formatting.

Lines 3200–3410 control the saving of the transactions to disk and exit the program. All the elements of the array CODE\$ are written to disk, followed by the value of NR (the number of records) and then the nonblank elements of the array RD\$.

Variable	Function
A1	Temporary use
A\$	Input and temporary storage
B0\$	Null string
B1\$	Blank string
B20\$	String of 20 spaces
BS\$	Backspace character
CLRLN	Location of routine to clear line
CODE\$()	Array storing code descriptions
CS	Location of routine to clear screen from cursor
D\$	Ctrl-D
ESC\$	Escape key
FS\$	Forward space
I, J	Counters, indices
LD, LC, L1, L2, LP, LS, LM	Field lengths used for printing reports
MR	Maximum number of transactions
NC	Number of codes - 1
NR	Number of transactions
QU\$	Quote
R\$	Return key
RD\$	Current transaction
RD\$()	Array storing transactions
RN	Pointer to transaction
TR	Pointer to top transaction
T\$, Z\$	Temporary storage
DT\$, CK\$, CD(), PY\$, DC\$, AMT\$	Fields of current record

Your Personal Ledger

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```

61 100 CLRLN = - 868: REM CLEAR TO END OF LINE SUBRO
    UTINE
27 110 CS = - 958: REM CLEAR TO END OF SCREEN SUBROU
    TINE
89 120 B0$ = "":B1$ = " ":B20$ = "
    "
3A 130 D$ = CHR$ (4): REM CTRL-D
23 140 QU$ = CHR$ (34): REM QUOTE
A1 150 R$ = CHR$ (13): REM RETURN
28 160 BS$ = CHR$ (8): REM BACK SPACE
DE 170 FS$ = CHR$ (21): REM FORWARD SPACE
3B 180 ESC$ = CHR$ (27)
22 190 MR = 400: REM MAX NO OF RECORDS
B3 200 NC = 25: REM NO. OF CODES
F6 210 DIM CODE$(1,NC),RD$(MR)
7A 220 LD = 8:LC = 5:L1 = 8:L2 = 8:LP = 15:LS = 15:L
    M = 10: REM DEFAULT REPORT FORMATTING WIDTHS
4B 230 HOME
E0 240 Z$ = "YOUR PERSONAL LEDGER": GOSUB 3420
5D 280 VTAB 10
29 290 PRINT "PRESS      TO"
7A 300 PRINT "-----  ---"
C9 310 PRINT "  1      USE EXISTING LEDGER"
A9 320 PRINT "  2      SET UP NEW LEDGER"
D5 330 GET A$
FE 340 IF A$ = "1" THEN 380
FF 350 IF A$ = "2" THEN 540
9C 360 GOTO 330
24 370 GOTO 290
83 380 REM EXISTING LEDGER
58 390 HOME
89 400 Z$ = "PLACE YOUR DATA DISK IN DRIVE 1": GOSUB
    3420
E9 410 Z$ = "AND": GOSUB 3420
53 420 Z$ = "PRESS <RETURN> TO CONTINUE": GOSUB 3420
55 430 GOSUB 3450: REM CLEAR BOTTOM
56 440 GET A$: PRINT
79 450 PRINT D$"OPEN LEDGER.DATA"
DF 460 PRINT D$"READ LEDGER.DATA"
A1 470 FOR I = 0 TO 1: FOR J = 0 TO NC: INPUT CODE$(
    I,J): NEXT : NEXT
26 480 INPUT NR:TR = NR
19 490 IF NOT NR THEN 510
2F 500 FOR I = 1 TO NR: INPUT RD$(I): NEXT
32 510 PRINT D$"CLOSE LEDGER.DATA"
67 520 RN = 1: REM SET CURRENT RECORD NUMBER

```

```

77 530 GOTO 600: REM MAIN MENU
80 540 REM NEW LEDGER
52 550 HOME
AD 560 Z$ = "PUT FORMATTED DATA DISK IN DRIVE 1": GO
SUB 3420
F6 570 Z$ = "AND": GOSUB 3420
7C 580 Z$ = "PRESS <RETURN>": GOSUB 3420: GET A$
C5 590 NR = 0: NT = 0
50 600 REM MAIN MENU
48 610 HOME
FE 620 PRINT "PRESS", "TO"
67 630 PRINT "-----", "----"
EF 640 PRINT " 1", "EDIT TRANSACTION CODES"
5E 650 PRINT " 2", "ENTER TRANSACTION"
8F 660 PRINT " 3", "FIND/CHANGE TRANSACTION"
33 670 PRINT " 4", "SORT TRANSACTIONS"
8C 680 PRINT " 5", "PRINT REPORT"
F7 690 PRINT
88 700 PRINT "<ESC>", "SAVE CHANGES/EXIT"
61 710 GET A$: IF (A$ < "1" OR A$ > "5") AND A$ < >
ESC$ THEN 710
97 720 IF A$ = ESC$ THEN 3200: REM SAVE TO DISK AND
EXIT
9A 730 ON VAL (A$) GOTO 750, 1070, 1250, 2030, 2220
53 740 GOSUB 3450: REM SET SCREEN
CB 750 REM EDIT TRANSACTION CODES
47 760 HOME : Z$ = "EDIT TRANSACTION CODES": GOSUB 34
20
F4 770 PRINT
A0 780 Z$ = "WHICH TYPE CODE": GOSUB 3420
5C 790 Z$ = "PRESS 1 OR 2": GOSUB 3420
6D 800 GET A$: IF A$ < "1" OR A$ > "2" THEN 600: REM
MAIN MENU
D7 810 I = VAL (A$) - 1
70 820 HOME : Z$ = "EDIT TYPE " + A$ + " TRANSACTION
CODE": GOSUB 3420
2C 830 VTAB 8
29 840 PRINT "CODE", "DESCRIPTION"
2F 850 J = 0
5F 860 GOSUB 3450: REM CLEAR BOTTOM
14 870 PRINT "PRESS <-- OR --> FOR OTHER CODES"
8C 880 PRINT " <C> TO CHANGE"
27 890 PRINT " <ESC> FOR MENU"
B2 900 VTAB 10: HTAB 2: CALL CLRLN
6E 910 PRINT CHR$ ( ASC ("A") + J), CODE$(I, J)
C9 920 VTAB 10: GET A$
27 930 IF A$ = BS$ THEN J = J - 1: GOTO 980
25 940 IF A$ = FS$ THEN J = J + 1: GOTO 980
5F 950 IF A$ = ESC$ THEN 600: REM MAIN MENU
E8 960 IF A$ = "C" THEN 1010

```



```

A6 970 GOTO 920
77 980 IF J < 0 THEN J = NC
E7 990 IF J > NC THEN J = 0
CC 1000 GOTO 900
93 1010 REM EDIT CODE I,J
38 1020 GOSUB 3450:Z$ = "ENTER NEW DESCRIPTION": GOS
    UB 3420
11 1030 VTAB 10: HTAB 2: CALL CLRLN
DC 1040 PRINT CHR$ ( ASC ("A") + J),
36 1050 INPUT " ";CODE$(I,J)
76 1060 GOTO 860: REM NEXT CODE
B0 1070 REM ENTER TRANSACTION
85 1080 IF TR < MR THEN 1120
5C 1090 HOME
43 1100 Z$ = "THERE IS NO ROOM FOR ANY MORE RECORDS"
    : GOSUB 3420
02 1110 PRINT : GOSUB 3530: GOTO 600: REM MENU
42 1120 HOME
41 1130 Z$ = "ENTER TRANSACTION": GOSUB 3420
C8 1140 GOSUB 3540: REM ENTER DATE
89 1150 GOSUB 3660: REM ENTER CHECK NO.
67 1160 FOR I = 0 TO 1: GOSUB 3730: NEXT : REM ENTER
    CODES
C3 1170 GOSUB 3930: REM ENTER SOURCE/PAYEE
0E 1180 GOSUB 4000: REM ENTER DESCRIPTION
25 1190 GOSUB 4070: REM ENTER AMOUNT
68 1200 GOSUB 3450
75 1210 Z$ = "PRESS <RETURN> TO ACCEPT": GOSUB 3420
8B 1220 GOSUB 3480
39 1230 REM ACCEPT ENTRY
46 1240 NR = NR + 1:TR = TR + 1:RN = TR: GOSUB 4150
5E 1250 REM FIND/CHANGE TRANSACTION
E3 1260 IF NR = 0 THEN 600: REM MENU
62 1270 GOSUB 4190: REM RETRIEVE RECORD RN
21 1280 HOME :Z$ = "FIND/DISPLAY TRANSACTION": GOSUB
    3420
59 1290 GOSUB 3620: REM DISPLAY DATE
DD 1300 GOSUB 3720: REM DISPLAY CHECK NO.
A3 1310 FOR I = 0 TO 1: GOSUB 3920: NEXT : REM DISPL
    AY CODES
1F 1320 GOSUB 3990: REM DISPLAY SOURCE/PAYEE
EA 1330 GOSUB 4060: REM DISPLAY DESCRIPTION
4E 1340 GOSUB 4140: REM DISPLAY AMOUNT
81 1350 GOSUB 3450
33 1360 PRINT "<--"
88 1370 PRINT "<C>HANGE"
AA 1380 PRINT "<P>RINT"
6D 1390 PRINT "<E>ENTER"
53 1400 GET A$
A9 1410 IF A$ = "J" THEN 1490

```

```

E7 1420 IF A$ = BS$ THEN ST = - 1: J = 1: GOTO 1520
D0 1430 IF A$ = FS$ THEN ST = 1: J = 1: GOTO 1520
9B 1440 IF A$ = "C" THEN 1590
89 1450 IF A$ = "D" THEN 1810
C7 1460 IF A$ = "P" THEN 1930
C1 1465 IF A$ = "E" THEN 1070
EC 1470 IF A$ = ESC$ THEN 600
7A 1480 GOTO 1400
F6 1490 GOSUB 3450: INPUT "JUMP HOW MANY RECORDS (+/-)? "; A$
19 1500 J = INT ( VAL (A$)): ST = 1: IF J < 0 THEN ST = - 1
3A 1510 J = ABS (J): IF NOT J THEN 1350
0D 1520 FOR I = 1 TO J
6D 1530 RN = RN + ST
D4 1540 IF RN > TR THEN RN = 1
C9 1550 IF RN < 1 THEN RN = TR
D7 1560 IF RD$(RN) = B0$ THEN 1530
91 1570 NEXT I
CB 1580 GOTO 1250: REM FIND/EDIT MENU
E6 1590 REM CHANGE
26 1600 Z$ = "DATE": GOSUB 1760
8A 1610 IF A$ = "C" THEN GOSUB 3540
82 1620 Z$ = "CHECK NUMBER": GOSUB 1760
D3 1630 IF A$ = "C" THEN GOSUB 3660
83 1640 FOR I = 0 TO 1
A9 1650 Z$ = "CODE " + STR$ (I + 1): GOSUB 1760
9E 1660 IF A$ = "C" THEN GOSUB 3730
93 1670 NEXT I
E1 1680 Z$ = "SOURCE/PAYEE": GOSUB 1760
2B 1690 IF A$ = "C" THEN GOSUB 3930
CC 1700 Z$ = "DESCRIPTION": GOSUB 1760
69 1710 IF A$ = "C" THEN GOSUB 4000
0F 1720 Z$ = "AMOUNT": GOSUB 1760
F4 1730 IF A$ = "C" THEN GOSUB 4070
04 1740 GOSUB 4150: REM STORE RECORD
C3 1750 GOTO 1250: REM FIND/EDIT MENU
01 1760 GOSUB 3450: Z$ = "PRESS <C> TO CHANGE " + Z$:
    GOSUB 3420
52 1770 Z$ = "OR": GOSUB 3420
9B 1780 Z$ = "PRESS <RETURN> TO ACCEPT": GOSUB 3420
2A 1790 GET A$: IF A$ < > R$ AND A$ < > "C" THEN 1790
    0
E1 1800 RETURN
DA 1810 REM DELETE RECORD
32 1820 GOSUB 3450: Z$ = "PRESS <RETURN> TO DELETE":
    GOSUB 3420
44 1830 Z$ = "OR": GOSUB 3420
E5 1840 Z$ = "PRESS <ESC> TO ABORT DELETION": GOSUB
    3420

```

```

A6 1850 GET A$: IF A$ < > R$ AND A$ < > ESC$ THEN 18
    50
22 1860 IF A$ = ESC$ THEN 1250
3B 1870 RD$(RN) = B0$
9E 1880 NR = NR - 1
C7 1890 IF NR = 0 THEN 600
6F 1900 RN = RN + 1: IF RN > TR THEN RN = 1
4B 1910 IF RD$(RN) = B0$ THEN 1900
7C 1920 GOTO 1250
60 1930 REM PRINT RECORD RN
CD 1940 PRINT : PRINT D$"PR#1"
B2 1950 PRINT "DATE:",DT$
81 1960 PRINT "CHECK NUMBER:",CK$
76 1970 FOR I = 0 TO 1: PRINT "CODE ";I + 1, CODE$(I,
    CD(I)): NEXT I
59 1980 PRINT "SOURCE/PAYEE:",PY$
BF 1990 PRINT "DESCRIPTION:",DC$
64 2000 PRINT "AMOUNT:",AMT$
F2 2010 PRINT D$"PR#0"
AA 2020 GOTO 1250: REM FIND/EDIT MENU
D4 2030 REM SORT TRANSACTIONS
A4 2040 IF NR = 0 THEN 600
83 2050 I = 1:RN = 1
D7 2060 IF RD$(I) = B0$ THEN I = I + 1: GOTO 2060
51 2070 IF I < > 1 THEN RD$(I) = RD$(I):RD$(I) = B0$
4A 2080 IF NR = 1 THEN TR = 1: GOTO 600: REM MENU
FD 2090 J = 2: REM NEXT RECORD PLACE
9A 2100 FOR I = 2 TO NR
95 2110 IF RD$(J) = B0$ THEN J = J + 1: GOTO 2110
D4 2120 IF I < > J THEN RD$(I) = RD$(J):RD$(J) = B0$
31 2130 I1 = I - 1:I2 = I
BD 2140 IF I1 = 0 THEN 2180
BB 2150 IF RD$(I1) < RD$(I2) THEN 2180
16 2160 T$ = RD$(I1):RD$(I1) = RD$(I2):RD$(I2) = T$
9D 2170 I2 = I1:I1 = I1 - 1: GOTO 2140
1D 2180 J = J + 1
92 2190 NEXT I
B1 2200 TR = NR
A3 2210 GOTO 600: REM MENU
B5 2220 REM REPORT
45 2230 HOME :Z$ = "PRINT REPORT": GOSUB 3420
6B 2240 GOSUB 3450:Z$ = "PRESS <RETURN> TO ACCEPT DE
    FAULT FORMAT": GOSUB 3420
41 2250 Z$ = "OR": GOSUB 3420
7D 2260 Z$ = "PRESS <C> TO CHANGE FORMAT": GOSUB 342
    0
0B 2270 GET A$: IF A$ < > R$ AND A$ < > "C" THEN 227
    0
92 2280 IF A$ = R$ THEN 2490
BC 2290 REM CHANGE FORMAT

```



```

50 2300 HOME :Z$ = "PRESS RETURN TO ACCEPT EACH FIEL
    D": GOSUB 3420
33 2310 Z$ = "OR": GOSUB 3420
65 2320 Z$ = "ENTER NEW FIELD WIDTH": GOSUB 3420
87 2330 Z$ = "WIDTH = 0 ==> OMIT FIELD": GOSUB 3420
8E 2340 VTAB 8: CALL CS: PRINT "DATE",LD
F6 2350 INPUT "":A$: IF A$ < > B0$ THEN LD = VAL (A$
    ): GOTO 2340
C8 2360 VTAB 9: CALL CS: PRINT "CHECK NUMBER",LC
1F 2370 INPUT "":A$: IF A$ < > B0$ THEN LC = VAL (A$
    ): GOTO 2360
BA 2380 VTAB 10: CALL CS: PRINT "CODE 1",L1
25 2390 INPUT "":A$: IF A$ < > B0$ THEN L1 = VAL (A$
    ): GOTO 2380
AF 2400 VTAB 11: CALL CS: PRINT "CODE 2",L2
36 2410 INPUT "":A$: IF A$ < > B0$ THEN L2 = VAL (A$
    ): GOTO 2400
12 2420 VTAB 12: CALL CS: PRINT "SOURCE/PAYEE",LP
42 2430 INPUT "":A$: IF A$ < > B0$ THEN LP = VAL (A$
    ): GOTO 2420
31 2440 VTAB 13: CALL CS: PRINT "DESCRIPTION",LS
EA 2450 INPUT "":A$: IF A$ < > B0$ THEN LS = VAL (A$
    ): GOTO 2440
9F 2460 VTAB 14: CALL CS: PRINT "AMOUNT",LM
72 2470 INPUT "":A$: IF A$ < > B0$ THEN LM = VAL (A$
    ): GOTO 2460
A0 2480 GOSUB 3470
D0 2490 HOME :Z$ = "SELECTION": GOSUB 3420
72 2500 GOSUB 3450
55 2510 Z$ = "PRESS <RETURN> TO ACCEPT ALL DATES": G
    OSUB 3420
38 2520 Z$ = "OR": GOSUB 3420
A4 2530 Z$ = "PRESS <C> TO CHOOSE PARTICULAR DATES":
    GOSUB 3420
FB 2540 GET A$: IF A$ < > R$ AND A$ < > "C" THEN 254
    0
95 2550 D1$ = B0$: IF A$ = R$ THEN 2610
A1 2560 GOSUB 3450: PRINT "FROM:",: GOSUB 2590:D1$ =
    T$
C2 2570 PRINT "TO:",: GOSUB 2590:D2$ = T$
86 2580 GOTO 2610
E2 2590 T$ = B0$: PRINT "MONTH (MM): ":: GOSUB 2600
    : PRINT : PRINT "YEAR (YY): ":: GOSUB 2600
    0: PRINT :T$ = RIGHT$ (T$,2) + LEFT$ (T$,2):
    RETURN
70 2600 FOR II = 1 TO 2: GET A$: PRINT A$,:T$ = T$ +
    A$: NEXT : RETURN
78 2610 FOR I = 0 TO 1
55 2620 VT = 8:HT = 1: GOSUB 3460:Z$ = "CHOOSE CODE
    " + STR$ (I + 1): GOSUB 3420

```

```

EE 2630 GOSUB 3450:Z$ = "PRESS <RETURN> TO ACCEPT AL
      L CODES": GOSUB 3420
45 2640 Z$ = "OR": GOSUB 3420
DF 2650 Z$ = "PRESS <C> TO CHOOSE PARTICULAR CODE":
      GOSUB 3420
10 2660 GET A$: IF A$ < > R$ AND A$ < > "C" THEN 266
      0
D6 2670 C(I) = - 1: IF A$ = R$ THEN 2800
F3 2680 GOSUB 3450:Z$ = "<--,-->": GOSUB 3420
9E 2690 Z$ = "PRESS <RETURN> TO ACCEPT": GOSUB 3420
FA 2700 J = 0
DB 2710 VTAB 8: HTAB 1: CALL CLRLN: PRINT CHR$ ( ASC
      ("A") + J),CODE$(I,J)
D0 2720 GET A$: IF A$ < > BS$ AND A$ < > FS$ AND A$
      < > R$ THEN 2720
8B 2730 IF A$ = R$ THEN 2790
53 2740 IF A$ = BS$ THEN J = J - 1: GOTO 2760
1D 2750 J = J + 1
95 2760 IF J < 0 THEN J = NC
76 2770 IF J > NC THEN J = 0
8C 2780 GOTO 2710
7A 2790 C(I) = J
7C 2800 NEXT I
C0 2810 PRINT : PRINT D$"PR#1"
74 2820 A = 0
74 2830 PRINT "PERSONAL LEDGER REPORT": PRINT
B5 2840 IF D1$ < > B0$ THEN PRINT MID$ (D1$,3,2);"/"
      ; LEFT$ (D1$,2);" TO "; MID$ (D2$,3,2);"/";
      LEFT$ (D2$,2): PRINT
8C 2850 FOR I = 0 TO 1
2B 2860 IF C(I) >= 0 THEN PRINT "CODE ";I + 1;": "
      ;CODE$(I,C(I)): PRINT
98 2870 NEXT I
28 2880 IF LD THEN PRINT LEFT$ ("DATE" + B20$,LD);B1
      $;
ED 2890 IF LC THEN PRINT RIGHT$ (B20$ + "CHECK",LC);
      B1$;
8C 2900 IF L1 THEN PRINT LEFT$ ("CODE 1" + B20$,L1);
      B1$;
41 2910 IF L2 THEN PRINT LEFT$ ("CODE 2" + B20$,L2);
      B1$;
74 2920 IF LP THEN PRINT LEFT$ ("SOURCE/PAYEE" + B20
      $,LP);B1$;
77 2930 IF LS THEN PRINT LEFT$ ("DESCRIPTION" + B20$
      ,LS);B1$;
F0 2940 IF LM THEN PRINT RIGHT$ (B20$ + "AMOUNT",LM)
      ;B1$;
93 2950 PRINT : PRINT
8D 2960 RT = RN: FOR RN = 1 TO TR

```

```

EA 2970 IF RD$(RN) = B0$ THEN 3150: REM SKIP BLANK R
      RECORDS
3D 2980 GOSUB 4190: REM RETRIEVE RECORD
56 2990 IF D1$ = B0$ THEN 3020
12 3000 IF LEFT$(RD$,4) < D1$ THEN 3150
2E 3010 IF LEFT$(RD$,4) > D2$ THEN 3150
49 3020 IF C(0) < 0 THEN 3040
AF 3030 IF CD(0) < > C(0) THEN 3150
5A 3040 IF C(1) < 0 THEN 3060
3A 3050 IF CD(1) < > C(1) THEN 3150
C7 3060 IF LD THEN PRINT LEFT$(DT$ + B20$,LD);B1$;
77 3070 IF LC THEN PRINT RIGHT$(B20$ + CK$,LC);B1$;
FE 3080 IF L1 THEN PRINT LEFT$(CODE$(0,CD(0)) + B20
      $,L1);B1$;
B7 3090 IF L2 THEN PRINT LEFT$(CODE$(1,CD(1)) + B20
      $,L2);B1$;
DA 3100 IF LP THEN PRINT LEFT$(PY$ + B20$,LP);B1$;
4A 3110 IF LS THEN PRINT LEFT$(DC$ + B20$,LS);B1$;
E8 3120 IF LM THEN PRINT RIGHT$(B20$ + AMT$,LM);B1$
      ;
B1 3130 PRINT
63 3140 A = A + VAL (AMT$)
6F 3150 NEXT RN:RN = RT
AA 3160 A1 = INT (A):A1$ = STR$ (A1):A2$ = STR$ ( IN
      T (100 * (A - A1) + .1)): IF LEN (A2$) = 1 T
      HEN A2$ = "0" + A2$
B9 3170 PRINT : PRINT "TOTAL:  $";A1$;". ";A2$
78 3180 PRINT D$"PR#0": REM RESET OUTPUT TO MONITOR
C2 3190 GOTO 600: REM MENU
D0 3200 REM SAVE ENTRIES/EXIT
42 3210 HOME
6E 3220 Z$ = "PRESS <RETURN> TO SAVE CHANGES": GOSUB
      3420
91 3230 GOSUB 3480
82 3240 HOME :Z$ = "SAVING CHANGES": GOSUB 3420
99 3250 ONERR GOTO 3410
A6 3260 PRINT D$"OPEN LEDGER.DATA"
2B 3270 PRINT D$"WRITE LEDGER.DATA"
6C 3280 FOR I = 0 TO 1: FOR J = 0 TO NC: PRINT CODE$
      (I,J): NEXT : NEXT
84 3290 PRINT NR
8F 3300 IF NOT NR THEN 3360
82 3310 I1 = 1
A5 3320 FOR I = 1 TO NR
2E 3330 IF RD$(I1) = B0$ THEN I1 = I1 + 1: GOTO 3330
2D 3340 PRINT QU$;RD$(I1):I1 = I1 + 1
BD 3350 NEXT
28 3360 PRINT D$"CLOSE LEDGER.DATA"
DE 3370 POKE 216,0: REM RESET ONERR

```



```

BF 3380 HOME :Z$ = "PRESS <RETURN> TO EXIT": GOSUB 3
    420
AB 3390 GOSUB 3480
CC 3400 END
40 3410 POKE 216,0: GOTO 3200: REM ERROR TRAP -- BAC
    K TO CHOICE
EF 3420 REM CENTER DISPLAY
F5 3430 TB = 20 - INT ( LEN (Z$) / 2): IF TB < 1 THE
    N TB = 1
47 3440 PRINT TAB( TB)Z$: RETURN
03 3450 VT = 18:HT = 1
62 3460 HTAB HT: VTAB VT: CALL CS: HTAB HT: VTAB VT:
    RETURN
6E 3470 Z$ = "PRESS <RETURN> TO CONTINUE": GOSUB 342
    0
52 3480 Z$ = "OR": GOSUB 3420
8D 3490 Z$ = "PRESS <ESC> TO RETURN TO MENU": GOSUB
    3420
5E 3500 GET A$: IF A$ < > R$ AND A$ < > ESC$ THEN 35
    00
B4 3510 IF A$ = ESC$ THEN POP : GOTO 600: REM MENU
E5 3520 RETURN
0F 3530 VTAB 22:Z$ = "PRESS ANY KEY TO CONTINUE": GO
    SUB 3420: GET A$: PRINT : RETURN
97 3540 REM ENTER DATE YY/MM/DD
69 3550 REM VTAB 8
1D 3560 REM WIDTH 6
0F 3570 GOSUB 3450: PRINT "DATE: ";
2D 3580 DT$ = B0$
8F 3590 FOR J = 1 TO 2: GOSUB 3640: GOSUB 3640:DT$ =
    DT$ + "/": PRINT "/";: NEXT
FC 3600 GOSUB 3640: GOSUB 3640
79 3610 GOSUB 3450
A7 3620 VTAB 8: HTAB 1: CALL CLRLN: PRINT "DATE:",DT
    $: RETURN : REM DISPLAY DATE
EB 3630 RETURN
18 3640 GET A$: IF A$ < "0" OR A$ > "9" THEN 3640
3F 3650 DT$ = DT$ + A$: PRINT A$;: RETURN
55 3660 REM ENTER CHECK NUMBER
7B 3670 REM VTAB 9
17 3680 REM WIDTH 5
99 3690 GOSUB 3450
51 3700 INPUT "CHECK NUMBER: ";A$: GOSUB 3450
CA 3710 CK$ = RIGHT$ (B20$ + A$,5)
07 3720 VTAB 9: HTAB 1: CALL CLRLN: PRINT "CHECK NO:
    ",CK$: RETURN : REM DISPLAY CHECK NO.
F7 3730 REM ENTER CODES
07 3740 REM I=WHICH CODE (0 OR 1)
79 3750 REM VTAB 10 OR 11
0D 3760 REM WIDTH 1

```

```

93 3770 GOSUB 3450
C1 3780 Z$ = "CHOOSE CODE " + STR$ (I + 1): GOSUB 34
    20
38 3790 VTAB 22: Z$ = "PRESS <--,--> OR <RETURN> TO A
    CCEPT": GOSUB 3420
FD 3800 J = 0
35 3810 VTAB 20: CALL CLRLN: Z$ = CHR$ (ASC ("A") +
    J) + " " + CODE$(I,J): GOSUB 3420
65 3820 GET A$
6E 3830 IF A$ = BS$ THEN J = J - 1: GOTO 3870
6A 3840 IF A$ = FS$ THEN J = J + 1: GOTO 3870
87 3850 IF A$ = R$ THEN 3900
8E 3860 GOTO 3820
9C 3870 IF J < 0 THEN J = NC
7D 3880 IF J > NC THEN J = 0
96 3890 GOTO 3810
74 3900 REM ACCEPT CODE
A7 3910 CD(I) = J
ED 3920 VTAB 10 + I: HTAB 1: CALL CLRLN: PRINT "CODE
    "; I + 1; "; ", CODE$(I,CD(I)): RETURN
34 3930 REM ENTER SOURCE/PAYEE
60 3940 REM VTAB12
F6 3950 REM WIDTH 20
93 3960 GOSUB 3450
84 3970 INPUT "SOURCE/PAYEE: "; A$: GOSUB 3450
2C 3980 PY$ = LEFT$ (A$ + B20$, 20)
8C 3990 VTAB 12: HTAB 1: CALL CLRLN: PRINT "SOURCE/P
    AYEE: ", PY$: RETURN : REM DISPLAY SOURCE/PAYE
    E
37 4000 REM ENTER DESCRIPTION
53 4010 REM VTAB13
D9 4020 REM WIDTH 20
76 4030 GOSUB 3450
95 4040 INPUT "DESCRIPTION: "; A$: GOSUB 3450
8B 4050 DC$ = LEFT$ (A$ + B20$, 20)
CD 4060 VTAB 13: HTAB 1: CALL CLRLN: PRINT "DESCRIPT
    ION: ", DC$: RETURN
0A 4070 REM ENTER AMOUNT
7F 4080 REM VTAB14
50 4090 REM WIDTH 9
EA 4100 GOSUB 3450: INPUT "AMOUNT: "; A$: GOSUB 3450
0A 4110 A = VAL (A$): A1 = INT (A): A2 = INT (100 * (A
    - A1) + .1)
90 4120 A2$ = STR$ (A2): IF LEN (A2$) = 1 THEN A2$ =
    "0" + A2$
6F 4130 AMT$ = RIGHT$ (B20$ + STR$ (A1) + "." + A2$,
    9)
D5 4140 VTAB 14: HTAB 1: CALL CLRLN: PRINT "AMOUNT:"
    , AMT$: RETURN : REM DISPLAY AMOUNT
82 4150 REM STORE RECORD RN

```

```
C5 4160 RD$ = MID$ (DT$,7,2) + MID$ (DT$,1,2) + MID$  
      (DT$,4,2) + CK$ + CHR$ (CD(0) + ASC ("A"))  
      + CHR$ (CD(1) + ASC ("A")) + PY$ + DC$ + AMT  
      $  
B3 4170 RD$(RN) = RD$  
F6 4180 RETURN  
DF 4190 REM RETRIEVE RECORD RN  
A9 4200 RD$ = RD$(RN): REM RECORD DATA  
46 4210 DT$ = MID$ (RD$,3,2) + "/" + MID$ (RD$,5,2)  
      + "/" + MID$ (RD$,1,2): REM DATE  
D6 4220 CK$ = MID$ (RD$,7,5): REM CHECK NO.  
CF 4230 FOR II = 0 TO 1:CD(II) = ASC ( MID$ (RD$,12  
      + II,1)) - ASC ("A"): NEXT : REM CODES 1 & 2  
3A 4240 PY$ = MID$ (RD$,14,20): REM PAYEE/SOURCE  
6B 4250 DC$ = MID$ (RD$,34,20): REM DESCRIPTION  
65 4260 AMT$ = MID$ (RD$,54,9): REM AMOUNT  
F4 4270 RETURN
```


Dr. Disk

Alan H. Stein

The ability to read and edit individual bytes stored on your disk can come in handy more than once. This utility, with versions for both DOS 3.3 and ProDOS, allows you to read, change, and write to disk any block, sector, or track. Undeleting files, searching for lost passwords, and simply examining how files are stored are just some of the applications of "Dr. Disk."

It's often extremely useful and even interesting to examine and change the contents of disks on a sector, block, and byte level. "Dr. Disk" is a machine language program designed to let you do just that.

Dr. Disk is easy to use. And though it's a machine language program, you don't have to know anything about machine language programming to type it in or use it, because Dr. Disk is listed in MLX format. Using the "Apple MLX" machine language entry program found in Appendix C, you can enter the data quickly and without errors.

Before you begin entering Dr. Disk, you'll need to read the MLX article, understand the instructions for using it, and type in and save a copy. (You'll want to use it again for future machine language programs from other COMPUTE! publications.) When you're ready to begin entry, run MLX. You'll be asked for the starting and ending addresses of the data. Your responses will depend on the operating system you use since there are separate versions of Dr. Disk for DOS 3.3 (Program 1) and ProDOS (Program 2):

DOS 3.3 version:

Starting address: 2000

Ending address: 2727

ProDOS version:

Starting address: 2000

Ending address: 272F

After entering these addresses, follow the MLX instructions and enter the data for the appropriate version of Dr. Disk. When you've finished entering the data, save it on disk with the name DR.DISK.

To load, run, and initialize the program (either the DOS 3.3 or ProDOS version), simply type

BRUN DR.DISK

On Your Screen

The first thing you'll see on the screen will be a request for a printer slot. If you're like most Apple II users, you have your printer connected to slot 1—simply press the 1 key. For additional flexibility, Dr. Disk accepts any key response between 1 and 7. Pressing a number key outside this range informs Dr. Disk that no printer is connected.

As soon as a key is pressed, the screen clears except for a menu on the bottom. The menu remains on the screen except when the contents of a sector or block are being printed. This helps make using Dr. Disk a snap. Depending on which version of Dr. Disk you're using, DOS 3.3 or ProDOS, the menu should look like one of the following:

DOS 3.3

DRIVE 1	READ
TRACK 11	WRITE
SECTOR OF	EDIT
^S SLOT 6	^P PRINT
^T TOGGLE ASCII MSB ON	^X EXIT

ProDOS

SLOT 6	READ BLOCK
DRIVE 1	WRITE BLOCK
BLOCK 000	EDIT BLOCK
TOGGLE ASCII<-->HEX	ASCII MSB ON
^P PRINT BLOCK	^X EXIT

Note that the first letters of *DRIVE*, *TRACK*, *SECTOR*, *READ*, *WRITE*, and *EDIT* are in reverse.

Dr. Disk, DOS 3.3

Here are brief explanations of each of the Dr. Disk DOS 3.3 commands.

DRIVE is used to choose which drive you're accessing, 1 or 2. Obviously, this is necessary only when you have a multidrive system. When Dr. Disk starts, the default is drive 1. Press D to change it. The cursor then moves to the word *DRIVE*. Press the number of the drive you want to use.

TRACK represents, in hexadecimal, the track you want to access. A disk has 35 tracks. To access a specific track, press T. The cursor moves to the word *TRACK*, at which point you can enter the track you want. Note that you must enter it in hexadecimal. For instance, you would enter track 25 as 19.

SECTOR represents the sector you want to access. On a disk, each track contains 16 sectors. Change the sector by pressing S and the hexadecimal digit representing the sector you want. Dr. Disk defaults to track \$11, sector \$0F, since that sector contains the Volume Table of Contents (VTOC) for a DOS 3.3 disk.

SLOT indicates which slot your disk drive is connected to. Since most disk drives are connected to slot 6, Dr. Disk boots with that as a default. To change the slot, simply press the Control key and S (Ctrl-S). A cursor moves next to the word *SLOT*. Press the number of the new slot.

READ transfers the contents of a sector on the disk to memory. Simply press R, and whatever sector has been specified is read. If an error occurs, the message **ERROR—PRESS ANY KEY TO CONTINUE** is displayed on the bottom of the screen, and the sector reading aborts. The program continues as soon as any key is pressed.

When a sector is read from disk, the contents appear on the screen in both hexadecimal and ASCII formats. You'll see 16 lines, each containing eight bytes of information. The number of the first byte shows on the left, followed by the hexadecimal representation of the eight bytes, and finally the ASCII representation. The cursor blinks to the left of a byte and can be moved around the pattern by pressing either the arrow keys, the traditional I, J, K, M diamond (up, left, right, and down, respectively) or the Return key. Since a screen can display only half a sector at a time, when the cursor is moved past the last byte or before the first byte, the other half is automatically displayed.

WRITE writes a sector back to disk. This will generally be used after some changes have been made. Since you can cause irreversible damage by inadvertently writing to disk, this operation is double-checked before it's allowed to proceed. The message **ARE YOU SURE YOU WANT TO WRITE (Y/N)?** appears at the bottom of the screen. If any key other than Y is pressed, the operation is aborted. An error during a write operation is treated the same way as an error during a read. If

you try to write to an empty drive, you'll find the contents of the last sector you accessed displayed on the screen.

EDIT allows you to change the contents of a sector. To enter the edit mode, simply press E. The message *PRESS ^Z TO TERMINATE EDIT MODE* displays. While in edit mode, you can change the contents of the sector simply by typing. Only the display changes at this point—no physical changes are made to the disk unless and until the write command is used.

The edit mode actually has three submodes—hexadecimal, ASCII with the most significant bit on, and ASCII with the most significant bit off. These submodes are displayed on the line starting with ^T TOGGLE and can be chosen by pressing Ctrl-T until the desired mode is displayed. Note that you must do this *before* entering the edit mode.

If you're in one of the two ASCII modes, pressing any key other than Ctrl-Z changes the contents of one byte to reflect the ASCII code of that key. Pressing several keys in succession causes the contents of consecutive bytes to be altered. All changes are displayed immediately. The only difference between the two submodes is the way the most significant bit of each byte is treated. The only way to exit these edit modes is by pressing Ctrl-Z.

If you're in hex mode, you can change the contents of bytes by entering the hexadecimal values. Each time you press a *pair* of hexadecimal digits, the contents of a byte are changed. Exit this mode either by pressing Ctrl-Z or by entering an invalid digit (anything but 0-9 and A-F).

PRINT prints the contents of the sector currently on the screen. This command is activated by pressing Ctrl-P. One word of caution: If an incorrect slot has been designated for the printer, the program will hang, or stop operation. You'll have to BRUN Dr. Disk again.

EXIT sends you back to BASIC. When you're finished with Dr. Disk, simply press Ctrl-X. (Here's a tip that will come in handy sooner or later: If you exit from Dr. Disk accidentally, you can get back by entering the monitor. Type CALL 8192 and press Return.)

Dr. Disk, ProDOS

The commands in the ProDOS version of Dr. Disk are identical in most respects. The only exceptions are the replacement of the Track and Sector commands with the Block command

and a change in how Dr. Disk displays blocks after a read.

BLOCK represents, in hexadecimal, the block you want to access. Floppy disks have 280 blocks, but hard disks may have considerably more. To access a specific block, press B. The cursor moves down to the word *BLOCK*, at which point you can press, in sequence, the three digits representing the block you want. For example, to access block 002, you'd press B, 0, 0, and 2. You'll usually start at that block since it contains the disk's directory.

READ BLOCK operates just as **READ** does in Dr. Disk, DOS 3.3, except that the screen can display only one-fourth a block at a time. When the cursor is moved past the last byte or before the first byte, another quarter of the block is displayed automatically.

Dr. Disk is surprisingly easy to use. It refuses to accept invalid commands while letting you use the keyboard in either uppercase or lowercase. Keep in mind, however, that Dr. Disk can be dangerous since you can change the contents of a sector and perhaps inadvertently destroy a disk. You should use Dr. Disk only on disks for which you have a backup copy.

I've used Dr. Disk to learn about the structure of DOS 3.3 and ProDOS, to manipulate catalogs, and even to change machine language programs. It can be extremely useful as well as a lot of fun.

The Doctor at Work

To understand the utility of Dr. Disk, let's walk through three practical examples:

- Rediscovering a forgotten password for a *Bank Street Writer* file.
- Resurrecting a file which has been unintentionally deleted (under both ProDOS and DOS 3.3).
- Creating a disk that can simultaneously hold both DOS 3.3 and ProDOS files.

Rediscovering a forgotten password is easiest, so let's consider that first even though not everyone with an Apple uses *Bank Street Writer*. The process may be different for other programs which require a password, but the ideas here should be similar.

Rediscovering a Forgotten Password

To make the explanation as simple and unambiguous as possible, it's assumed that you have a freshly initialized DOS 3.3 disk. (You can initialize a disk either through the utilities contained in *Bank Street Writer* or by typing INIT HELLO after booting any standard DOS 3.3 disk.)

Use *Bank Street Writer* to save a short text file. Call the file COMPUTE and give it the password TEST. Pretend that you forgot the password. *Bank Street Writer* makes it impossible to retrieve a file saved with a password unless you use that password. If you try to look at a catalog from within *Bank Street Writer* (or even outside *Bank Street Writer*), all you'll see is the word COMPUTE. Fortunately, there's a way to retrieve that password—Dr. Disk (DOS 3.3 version, obviously).

Run Dr. Disk. Conveniently, the first track/sector combination listed when you run the program is track \$11, sector \$0F, which happens to be precisely where the start of the disk directory is stored on a standard DOS 3.3 disk. Make sure the disk with the *Bank Street Writer* file is in the drive. All you have to do is press R (for READ) to read the contents of the sector from the disk. Part way down the right-hand column, the word COMPUTE will stand out. Right next to it will be the word TEST, your forgotten password. You can now access the file again.

Why was it so easy to find the password? DOS 3.3 stores the names, locations, and other information about its files in a directory which begins on track \$11, sector \$0F, and continues on track \$11 through sectors \$0E, \$0D, \$0C, and so on. You can easily pick out the names of files by looking through those sectors. If the COMPUTE file had not been in sector \$0F, you could just have looked further.

Why did you see the password TEST when you looked through the sector using Dr. Disk, but not when you asked for a catalog? The answer can be found by looking at the hexadecimal codes for the characters in the words COMPUTE and TEST. The codes are different, even for the same letters. That's because the characters in the password are stored as control characters rather than ordinary characters. For example, the S in the password TEST is stored as Ctrl-S, hexadecimal code \$93, rather than as S, hexadecimal code \$D3. Since control characters are not displayed when a disk is cataloged, the

password is invisible. It's actually part of the file's name, however, and is easily seen when examining the directory sector with Dr. Disk.

Undeleting a File Under DOS 3.3.

To see how to undelete a file using Dr. Disk, DOS 3.3, we'll assume that you have a newly initialized DOS 3.3 disk with a single file on it—an Applesoft program named HELLO which prints out THIS IS A TEST FOR COMPUTE! when executed.

Use Dr. Disk to examine track \$11, sector \$0F, of the sample disk. If you look at the ASCII characters on the right, you'll see the word *HELLO* starting at byte \$0E. If you look at the hexadecimal codes for the three bytes (\$0B, \$0C, and \$0D) preceding the beginning of the filename, you'll see 12 0F 02. These give the track (\$12) and sector (\$0F) of the track/sector list for the file along with a code (\$02) for the file type (Applesoft). After a series of \$A0's (the code for a blank), you'll see 02, which stands for the length of the file in sectors.

Exit from Dr. Disk by pressing Ctrl-X and look at the disk's catalog by typing CATALOG. You should see one file listed, the HELLO program. Delete that file by typing DELETE HELLO. If you look at the catalog again, no files will show. Now go back to Dr. Disk. (This can be done in two ways—Dr. Disk can be BRUN again, or you can type CALL 8192 and hit Return. The latter method will work as long as nothing has been done to disturb the image of Dr. Disk in memory.)

If you now examine the same sector of the sample disk, you'll see exactly two changes. Byte \$0B has been changed from 12 to FF to signify a deleted file. And byte \$2B has been changed from 00 to 12. (This byte comes just before the byte containing the file length.) All you have to do to get HELLO back into the catalog is change those two bytes back to their original states (easy) and update the VTOC (Volume Table of Contents—not so easy).

To change the two bytes in this sector, first move the cursor to byte \$0E by using either the arrow keys or the I-J-K-M key diamond. Next, perform the following steps:

- Press Ctrl-T to get to hex mode for input.
- Press E to enter edit mode.
- Press 1 and then 2 to change byte \$0B to 12.
- Press Ctrl-Z to get out of the edit mode.
- Move the cursor to byte \$2B.

- Press E to reenter edit mode.
- Press A and then 0 to change byte \$2B to A0.
- Press Ctrl-Z to get out of the edit mode.
- Press W to write the sector to disk.
- Press Y to confirm that you want to write to disk.

You've just put HELLO back into the catalog. You could now exit from Dr. Disk and see HELLO in the catalog and even execute it; however, that would be living dangerously. Here's why: DOS 3.3 uses a special table called the VTOC free-sector bitmap to determine which sectors are in use. The VTOC has not yet been updated. If you don't update the VTOC and then try to save another file, the first file may be overwritten, creating garbage. There are two ways to update the VTOC. One is routine, but very slow; the other is much faster, but also much more involved.

The routine but slow method consists of taking an initialized disk and using *FID* or a similar utility to copy every file of your original disk to the new disk. *FID* will automatically take care of the VTOC, but every file must be transferred individually, and the original disk must be reinitialized.

The quicker way is to use Dr. Disk again. To use this method to undelete any file, you need some understanding of how DOS 3.3 stores the directory. Also, if you want to reconstruct the VTOC manually rather than by recopying each file to a separate disk, details of the VTOC are necessary. The following explanation should suffice. Apple's reference manuals and the excellent reference *Beneath Apple DOS*, by Don Worth and Pieter Lechner, can supply more detailed information.

How DOS 3.3 Stores the Directory

DOS 3.3 stores its directory, as mentioned earlier, starting on track \$11, sector \$0F. On each directory sector, bytes \$01 and \$02 (the second and third bytes, since the bytes are numbered from \$00) contain a *link* to the next directory sector. Byte \$01 contains the track, while byte \$02 contains the sector. The actual directory entries begin with byte \$0B. Each entry contains \$23 (35 in decimal, but it's more convenient to think in hexadecimal) bytes. Let's number these bytes \$00-\$22, remembering that this

is relative to the start of the entry rather than the start of the sector.

Byte \$00 of the entry contains the track of the *track/sector list*, while byte \$01 contains the sector of that list. Byte \$02 contains a code for the file type. The name of the file begins at byte \$03 and continues through byte \$20, a total of 30 (decimal) bytes, which is the maximum length of a filename. It's generally easiest to recognize the start of the filename and work from there to find the other fields. Finally, bytes \$21-\$22 contain the length of the file. The only bytes that concern us are bytes \$00, \$01, and \$20. Indeed, the only bytes changed in the directory when a file is deleted are bytes \$00 and \$20. The contents of byte \$00 are transferred to byte \$20 and replaced by \$FF. DOS 3.3 recognizes a deleted file simply by finding \$FF at byte \$00. All you have to do to the directory entry to undelete the file is find the contents of byte \$20 and copy them to byte \$00. Since byte \$20 actually contains the last character of the filename, which usually is a blank, it's good practice to change byte \$20 to \$A0, the code for a blank, as well.

Updating the VTOC is considerably more complicated, so you may want to skip the next section and update the VTOC by copying each file to a fresh disk. (An alternative and perhaps faster method would be to copy the undeleted file to a separate disk, delete it again from the original disk, and then recopy it from the separate disk to the original disk.)

The VTOC resides on track \$11, sector \$00, and contains a bitmap of free sectors. The status of each track is contained in a group of four bytes. These groups of bytes start at byte \$38 and continue through byte \$C3. Bytes \$38-\$3B contain the status of track \$00, bytes \$3C-\$3F contain the status of track \$01, bytes \$40-\$43 contain the status of track \$02, and so on, continuing to bytes \$C0-\$C3, which contain the status of track \$22.

Within each group of four bytes, only the first two are used; the last two are set to \$00. The first two bytes contain a total of \$10 (16 in decimal) bits, one for each sector on the track. Each bit set to 1 corresponds to a sector which is free, while each bit set to 0 corresponds to a

sector which is in use. The bits of the second byte, from low bit to high bit, represent sectors \$0-\$7, while the bits of the first byte, also from low bit to high bit, represent sectors \$8-\$F. This may seem backward, but the reason should become clear from the following example.

Suppose sectors \$02, \$07, \$0C, and \$0F are free on track \$03, while all other sectors of track \$03 are in use. Write down 16 (\$10) binary digits next to each other, numbering them from \$00 to \$0F, right to left:

0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03 02 10 00

Set digits \$02, \$07, \$0C, and \$0F to 1, and all the others to 0. You should get the following binary number:

0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03 02 10 00
1 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0

Now convert that binary number to hexadecimal. You should get \$9084. In the VTOC free-sector bitmap, bytes \$44-\$47 correspond to track \$03. Byte \$44 would contain \$90, byte \$45 would contain \$84, while bytes \$46 and \$47 would each contain \$00.

In order to update the VTOC when undeleting a file, you must determine which sectors are used by the file and clear the corresponding bits in the free-sector bitmap to zero.

To determine which sectors are used by a given file, refer to its track/sector list. As described earlier, the location of the first sector in the track/sector list is pointed to by bytes \$00-\$01 of the file's directory entry. The following is a brief description of the relevant portions of each sector of the track/sector list.

Bytes \$01 and \$02 (the second and third bytes, remember) point to the next sector in the track/sector list. The last sector of the track/sector list contains \$00 in these bytes. Then, starting with byte \$0C, each pair of bytes gives the track and sector of a sector used in the file. Bytes not needed to give the locations of sectors in use contain \$00.

Thus, you have the following process for updating the VTOC. First, you must look through the track/sector list and make a list of all sectors used by the file. (Don't forget that the sectors containing the track/sector list are

themselves in use and must be included in the list.) You must then clear each corresponding bit in the free-sector bitmap to zero. This is a tedious process, prone to error. On occasion, I've found it fun to do, but it's generally much safer to recopy the files and let FID do the update.

Undeleting a File Under ProDOS

The process of undeleting a file under ProDOS bears some similarities to the corresponding process under DOS 3.3. Let's go through another simple example.

Again, you should have a formatted ProDOS disk, with the volume name /COMPUTE, containing a single short Applesoft program named TEST. This may be created by formatting a disk using the ProDOS filer and then saving a short, or even blank, program under the name TEST.

Execute Dr. Disk (ProDOS version) and put in the sample disk. Look at block \$002 by pressing B 002 R (B for block, 002 for block \$02, R for read block). The relevant parts are bytes \$05-\$13, containing the volume directory name COMPUTE; byte \$25, containing \$01, representing the fact that the /COMPUTE directory contains one file (TEST); bytes \$2C-\$3A, containing the filename TEST; byte \$2B, containing \$14, which shows that this is a *seedling* file with a filename \$04 bytes long; and byte \$3C, containing \$07, which shows that the file's *key block* is block \$007.

Exit Dr. Disk by pressing Ctrl-X. You can type CAT to see the catalog, which will contain the single file TEST. (Recall that ProDOS commands can be entered from the monitor, with the * prompt, and that you don't have to reenter BASIC.) Now delete the file by typing DELETE TEST. You can check that TEST is no longer in the catalog. Reenter Dr. Disk by typing 2000G (from the monitor) and read in block \$002 again.

You'll see the following changes: Byte \$25, giving the file count, has been changed to \$00 since there are no longer any files in the directory, while byte \$2B, giving the storage type and name length, has been changed to \$00 to indicate that the file TEST has been deleted.

You can undelete TEST by moving the cursor to byte \$25 and pressing T, E, 0, and 1, then Ctrl-Z to change byte \$25 back to \$01. Change byte \$2B back to \$14 in the same way.

You must, of course, remember to write the changes back to disk by pressing W and then Y.

You also have to update the Volume Bit Map, the ProDOS counterpart of DOS 3.3's VTOC free-sector bitmap. As with DOS 3.3, this can be done in two ways. You can use the ProDOS *Filer* program the same way you used the DOS 3.3 *FID* program and let *Filer* figure out what to do with the Volume Bit Map, or you can update the bitmap directly. In this particular case, you can update the bitmap by changing byte \$00 of block \$006 to \$00. (In general, since updating the ProDOS Volume Bit Map is a more convoluted process than updating the DOS 3.3 VTOC free-sector bitmap, I advise using *Filer*.)

As with DOS 3.3, to undelete arbitrary files, you must know a little more about the structure of ProDOS. You can find more detailed information in Apple's *ProDOS Technical Manual* or in *Beneath Apple ProDOS*.

ProDOS File Structure

Starting with block \$002 and usually continuing on consecutive blocks, ProDOS contains *directory headers* and *file descriptive entries*. Bytes \$00-\$03 of each block contain links to similar blocks. The important parts of a directory header, at least for our purposes, are the *name* and the *file count*. The name is important because you'll usually be able to recognize the name of a directory and from that name locate the file count, which occurs \$20 (32 decimal) bytes after the start of the directory name. Thus, when you undelete a file from a given directory, you must first find the directory name and then look \$20 bytes further on. Conveniently, this is exactly four rows below the first character of the filename on the Dr. Disk display. (One complication: We recognize the filename from its ASCII characters, which may flash or display in reverse on our display, while we manipulate the hexadecimal codes for the file count.) Just increase the byte containing the file count by \$01. (One more complication: If the byte contains \$FF, change it to \$00 and then increase the next byte by \$01. This is because the file count is actually stored in two bytes. You'll rarely have to be concerned with the upper byte, however.)

The relevant parts of the file descriptive entry are its *name*, *storage type*, and *name length*. The name is easy to find and doesn't change. The byte before the name contains both the storage type and name length. That byte changes to \$00 when a file is deleted and must be changed back when the file is undeleted.

The contents of the byte may be split into two hexadecimal digits. The second digit represents the length of the filename. That can easily be determined by looking at the filename and counting up its characters, remembering of course to count in hexadecimal. The first digit represents the storage type. Generally, that digit will be either \$1, \$2, or \$3, corresponding to a *seedling*, *sapling*, or *tree file*. These files represent Apple trees in various stages of growth. Actually, a seedling file is a file containing just 1 data block, a sapling is a file containing from 2 to 256 data blocks, and a tree is a file containing from 257 to 32,768 data blocks. To resurrect a deleted file, therefore, you must remember, roughly, a file's length. (Most files will probably be saplings, in the range of 2 to 256 data blocks.)

In our example, the file TEST used just one block; hence, it was a seedling file and the first digit was 1. The length of the name TEST was 4, so the second digit was 4. The byte immediately before the name, then, should be changed to \$14.

If you remember the approximate size of a deleted file, the ProDOS undeletion process is a simple two-step process: Increase the file count of the directory header by \$01 and replace the byte containing the storage type and the name length in the file descriptive entry. Of course, don't forget to use FILER to tediously update the Volume Bit Map.

Creating a Disk Which Can Contain Both ProDOS and DOS 3.3 Files

This process of creating a disk which can store both ProDOS and DOS 3.3 files is simpler than you might think. This is because ProDOS and DOS 3.3 keep their vital information in different places. Furthermore, once you create one such disk, you

never have to go through the process again, since you can use any standard copying program, such as *COPYA* or the ProDOS Copy command in *Filer*, to duplicate your prototype.

(Note: The process described here is not original. I've found it in several places, including CompuServe. Naturally, I didn't learn that until I had gone through the painful process of figuring it out myself.)

To create your double-DOS disk, follow these steps:

1. Format a ProDOS disk.
2. Initialize a DOS 3.3 disk and delete the HELLO program.
3. Copy track \$11 from the DOS 3.3 disk to the ProDOS disk. This is the track that contains the VTOC and catalog sectors. This can be done in any one of the following three ways.

Using Dr. Disk, DOS 3.3 version. Successively copy each sector, \$00-\$0F, from the DOS 3.3 disk to the ProDOS disk as follows.

- Boot Dr. Disk and press T, 1, and 1.
- Put the DOS 3.3 disk in your drive. Press S, 0, and R (this will read sector 0).
- Take out the DOS 3.3 disk and put in the ProDOS disk.
- Press W and Y (this will write sector 0 to the ProDOS disk).
- Repeat this process for sectors \$01, \$02, and so on, to sector \$0F.

This can be speeded up if you have two drives, in which case, rather than swapping disks, you can alternately specify which drive you want to read from or write to.

Using Dr. Disk, ProDOS version. Successively copy blocks \$088-\$08F from the DOS 3.3 disk to the ProDOS disk. These are the blocks, according to ProDOS, which correspond to DOS 3.3 track \$11. Follow these directions:

- Boot Dr. Disk, ProDOS version.
- Put the DOS 3.3 disk in your drive.
- Press B, 0, 8, 8, and R (this reads block \$088).
- Replace the DOS 3.3 disk with the ProDOS disk.
- Press W and Y (this writes block \$088 to the ProDOS disk).

Repeat the process for blocks \$089, \$08A, and so on, until block \$08F. As with the process using the DOS 3.3

version of Dr. Disk, you can do this without disk swapping if you have two drives. Clearly, it's faster to copy the track using the ProDOS version of Dr. Disk rather than the DOS 3.3 version.

Copy utility. The quickest method of all is to use a program which has the capability of copying individual tracks. (But you may not have such a utility, in which case Dr. Disk will be invaluable.)

4. Discard the DOS 3.3 disk. Everything from here on refers to the disk that originally contained only ProDOS.
5. Change the Volume Bit Map to make ProDOS think that everything from block \$088 on is used. This can be done using either version of Dr. Disk. If you're using the ProDOS version, you'll be dealing with block \$006 (press B, 0, 0, 6, and R); if you're using DOS 3.3, you'll be dealing with track \$00, sector \$03 (press T, 0, 0, S, 3, and R).

With either Dr. Disk, change the contents of bytes \$11-\$22 to 00's. Make sure you're in hex mode before you enter the editing mode. Press W and Y when you're finished to write the changes to disk.

6. Change the DOS 3.3 VTOC free-sector bitmap to make DOS 3.3 think that every sector in tracks \$00-\$10 is used. If you're using the ProDOS version of Dr. Disk, you'll be dealing with block \$88, while if you're in the DOS 3.3 version of Dr. Disk, you'll be dealing with track \$11, sector \$00.

With either version, change the contents of bytes \$44-\$7C to 00's and make sure you write the changes to disk.

You now have a disk that can hold both types of files. If you're using ProDOS, files will be saved only to blocks \$000-\$087, while if you're using DOS 3.3, files will be saved only to tracks \$11-\$22. In either case, you have fooled the operating system into thinking the rest of the disk is already written to, which it very well may be. If you analyze what we've done more carefully, you'll realize that the way we tampered with the Volume Bit Map and the VTOC was somewhat arbitrary. Different portions of the disk could be allocated to each system.

As it stands, the disk is unbootable. However, due to the nature of ProDOS, it's a simple matter to make it boot ProDOS while still retaining its dual-DOS capacity. All you need to do is copy two files to it: PRODOS and BASIC.SYSTEM.

The disk would then act like an ordinary bootable ProDOS disk if booted, but it would have a limited storage capacity since about half the disk was marked off-limits. But it would retain the ability to store DOS 3.3 files.

Program 1. Dr. Disk, DOS 3.3

For mistake-proof program entry, use "AppleMLX" (Appendix C) to type in this program.

START ADDRESS: 2000

END ADDRESS: 2727

```

2000: 20 58 FC A9 28 85 F9 A9 95
2008: 25 85 FA 20 36 24 20 D4 F4
2010: 24 C9 01 90 09 C9 08 B0 2E
2018: 05 2C 1E 27 10 02 A9 80 78
2020: 8D 22 27 A9 06 8D 1B 27 F2
2028: A9 01 8D 1A 27 A9 11 8D 60
2030: 1C 27 A9 0F 8D 1D 27 38 D5
2038: 6E 23 27 A9 00 85 FB 8D 93
2040: 18 27 8D 1F 27 8D 21 27 D2
2048: A9 10 85 FC A9 FF 8D 20 6A
2050: 27 20 58 FC A9 A5 85 F9 EF
2058: A9 25 85 FA A9 11 85 25 D8
2060: A9 00 85 24 20 3E 24 2C D6
2068: 23 27 30 03 4C 05 24 2C 25
2070: 21 27 10 03 4C 9B 22 A9 FB
2078: 11 85 25 20 95 24 A0 0A D1
2080: 84 24 88 AD 1A 27 09 B0 28
2088: 91 28 A9 14 85 25 20 95 A8
2090: 24 AD 1B 27 09 B0 91 28 7A
2098: A9 12 85 25 20 95 24 AD 82
20A0: 1C 27 C6 24 20 DA FD A9 E5
20A8: 13 85 25 20 95 24 AD 1D 30
20B0: 27 C6 24 C6 24 20 DA FD 7C
20B8: 2C 1F 27 30 1B 2C 20 27 AF
20C0: 30 0B A9 71 85 F9 A9 26 B5
20C8: 85 FA 4C E0 20 A9 64 85 18
20D0: F9 A9 26 85 FA 4C E0 20 80
20D8: A9 7F 85 F9 A9 26 85 FA 0A
20E0: A9 0A 85 24 A9 15 85 25 3D
20E8: 20 3E 24 A9 00 85 24 A9 EF
20F0: 17 85 25 20 95 24 20 9C DE
20F8: FC 20 C3 23 20 B6 24 20 AE
2100: BE 24 A0 00 D9 14 25 F0 19
2108: 09 C8 BE 14 25 D0 F5 4C BE
2110: FF 20 98 48 A4 24 88 A9 62
2118: A0 91 28 68 0A A8 C8 B9 D8
2120: EE 24 48 88 B9 EE 24 48 8E
2128: 60 A9 0A 85 24 A9 14 85 14

```

```

2130: 25 20 B3 24 20 D4 24 C9 2C
2138: 08 B0 08 2C 1E 27 30 03 5F
2140: 8D 1B 27 4C 6F 20 A9 0A 13
2148: 85 24 A9 11 85 25 20 B3 51
2150: 24 20 D4 24 F0 0C C9 03 D7
2158: B0 08 2C 1E 27 30 03 8D E9
2160: 1A 27 4C 6F 20 A9 0A 85 3B
2168: 24 A9 12 85 25 20 B3 24 F6
2170: 20 9D 24 2C 1E 27 30 EA 4A
2178: 8D 1C 27 4C 6F 20 A9 0A 8B
2180: 85 24 A9 13 85 25 20 B3 A9
2188: 24 20 D4 24 2C 1E 27 30 1A
2190: D1 8D 1D 27 4C 6F 20 A9 3F
2198: 01 8D 24 27 18 6E 23 27 9D
21A0: 20 E0 21 A9 00 8D 18 27 77
21A8: 4C 51 20 20 80 FE A9 AD 6C
21B0: 85 F9 A9 26 85 FA A9 00 37
21B8: 85 24 A9 17 85 25 20 3E AC
21C0: 24 20 84 FE A0 26 A9 60 EE
21C8: 91 28 20 BE 24 C9 D9 D0 9A
21D0: 0C A9 02 8D 24 27 20 E0 7B
21D8: 21 18 6E 23 27 4C 6F 20 1B
21E0: 20 E3 03 84 00 85 01 AD 9A
21E8: 1B 27 0A 0A 0A 0A A0 01 1F
21F0: 91 00 AD 1A 27 A0 02 91 A4
21F8: 00 A9 00 A0 03 91 00 AD BB
2200: 1C 27 A0 04 91 00 AD 1D 75
2208: 27 A0 05 91 00 A5 FB A0 F0
2210: 08 91 00 A5 FC C8 91 00 45
2218: AD 24 27 A0 0C 91 00 20 F1
2220: E3 03 20 D9 03 A9 00 85 FC
2228: 48 90 1F 6E 23 27 A9 ED 76
2230: 85 F9 A9 26 85 FA A9 00 8B
2238: 85 24 A9 17 85 25 20 80 70
2240: FE 20 3E 24 20 84 FE 20 47
2248: C8 24 60 AD 18 27 38 E9 98
2250: 08 8D 18 27 09 87 4C 61 D1
2258: 22 CE 18 27 AD 18 27 09 FB
2260: 80 49 FF D0 1E 4C 51 20 29
2268: AD 18 27 18 69 08 8D 18 8E
2270: 27 29 78 4C 7C 22 EE 18 C8
2278: 27 AD 18 27 29 7F D0 03 1D
2280: 4C 51 20 4C 6F 20 AD 1F 7E
2288: 27 49 80 8D 1F 27 30 08 99
2290: AD 20 27 49 80 8D 20 27 CE
2298: 4C 6F 20 A9 80 8D 21 27 21
22A0: 20 C3 23 20 B6 24 A9 00 E5
22A8: 85 24 A9 17 85 25 A9 8D 01
22B0: 85 F9 A9 26 85 FA 20 3E 64
22B8: 24 2C 1F 27 30 12 A9 00 8D

```

```

22C0: 8D 1E 27 20 C8 24 C9 9A 3F
22C8: F0 22 2D 20 27 4C D3 22 E9
22D0: 20 9D 24 48 A9 A0 A4 24 D2
22D8: 88 91 28 68 2C 1E 27 30 A9
22E0: 0B AC 18 27 91 FB 20 E8 F0
22E8: 23 4C 76 22 A9 00 8D 21 4C
22F0: 27 4C 6F 20 2C 22 27 10 14
22F8: 03 4C 6F 20 20 89 FE AD 94
2300: 22 27 20 95 FE A5 FB 85 8A
2308: FD A5 FC 85 FE 20 E2 23 10
2310: AD 1C 27 29 F0 18 6A 6A D2
2318: 6A 6A 09 B0 8D DC 26 AD 34
2320: 1C 27 29 0F C9 0A 30 0C 37
2328: 20 2E 23 4C 36 23 38 E9 CB
2330: 09 09 C0 60 09 B0 8D DD 5F
2338: 26 AD 1D 27 29 0F C9 0A 36
2340: 30 06 20 2E 23 4C 4A 23 09
2348: 09 B0 8D EA 26 A9 D3 85 A4
2350: F9 A9 26 85 FA 20 3E 24 14
2358: A9 00 85 F7 A9 80 8D 1E 2C
2360: 27 A5 F7 D0 08 2C 1E 27 04
2368: 10 4A 8D 1E 27 20 DA FD 4A
2370: A9 A0 20 ED FD 20 ED FD E0
2378: A0 00 B1 FD 20 DA FD A9 37
2380: A0 20 ED FD C8 C0 10 D0 F6
2388: F1 20 ED FD A0 00 B1 FD D3
2390: 29 7F C9 20 B0 02 A9 AE 16
2398: 20 ED FD C8 C0 10 D0 EE 8D
23A0: 20 E2 23 18 A5 FD 69 10 9D
23A8: 85 FD 18 A5 F7 69 10 85 99
23B0: F7 4C 61 23 20 93 FE 20 D1
23B8: 51 A8 4C 51 20 20 58 FC 9F
23C0: 4C D0 03 AD 18 27 29 07 53
23C8: 85 F9 0A 65 F9 69 06 85 EE
23D0: 24 AD 18 27 29 78 4A 4A 14
23D8: 4A 18 69 00 85 25 20 95 0E
23E0: 24 60 A9 8D 20 ED FD 60 74
23E8: 20 C3 23 A4 24 88 A9 A0 16
23F0: 91 28 AC 18 27 B1 FB 48 61
23F8: 20 DA FD 18 A5 F9 69 20 4F
2400: A8 68 91 28 60 AD 18 27 7C
2408: 48 29 80 8D 18 27 20 C3 09
2410: 23 A9 00 85 24 AD 18 27 DB
2418: 20 DA FD 20 E8 23 EE 18 B2
2420: 27 AD 18 27 29 07 D0 F3 D7
2428: AD 18 27 29 7F D0 DF 68 2C
2430: 8D 18 27 4C 6F 20 A9 80 BE
2438: 8D 1E 27 4C 43 24 A9 00 76
2440: 8D 1E 27 20 95 24 A0 00 3C
2448: B1 F9 8D 15 27 2C 1E 27 38

```



```

2450: F0 2A 8C 16 27 A9 14 85 1C
2458: 24 A9 00 8D 17 27 AD 17 BD
2460: 27 49 80 8D 17 27 30 02 2F
2468: C6 24 C8 B1 F9 CD 15 27 A9
2470: F0 07 C9 8D F0 03 4C 5E 8F
2478: 24 AC 16 27 20 85 24 CD 60
2480: 15 27 D0 C9 60 C8 B1 F9 57
2488: CD 15 27 F0 07 20 ED FD 83
2490: C9 8D D0 F1 60 48 A5 25 EE
2498: 20 C1 FB 68 60 20 D4 24 B8
24A0: 2C 1E 27 30 0D 0A 0A 0A 1D
24A8: 0A 8D 16 27 20 D8 24 0D 48
24B0: 16 27 60 20 95 24 A4 24 86
24B8: 88 A9 60 91 28 60 20 C8 A0
24C0: 24 C9 E0 90 02 29 DF 60 87
24C8: 2C 00 C0 10 FB AD 00 C0 97
24D0: 2C 10 C0 60 18 6E 1E 27 2F
24D8: 20 C8 24 49 B0 C9 0A 90 CD
24E0: 0A 69 88 C9 FA B0 04 38 11
24E8: 6E 1E 27 29 0F 60 45 21 0D
24F0: 28 21 64 21 7D 21 96 21 F2
24F8: AA 21 85 22 9A 22 4A 22 C5
2500: 4A 22 58 22 58 22 75 22 7D
2508: 75 22 67 22 67 22 67 22 59
2510: F3 22 BC 23 C4 93 D4 D3 98
2518: D2 D7 94 C5 C9 8B CA 88 4B
2520: CB 95 CD 8A 8D 90 98 00 F7
2528: AA D7 C8 C1 D4 A0 D3 CC 90
2530: CF D4 A0 C9 D3 A0 D9 CF EC
2538: D5 D2 A0 D0 D2 C9 CE D4 73
2540: C5 D2 8D C3 CF CE CE C5 2D
2548: C3 D4 C5 C4 A0 D4 CF BF 66
2550: 8D 8D C1 A0 D2 C5 D3 D0 2D
2558: CF CE D3 C5 A0 CF D5 D4 D9
2560: D3 C9 C4 C5 A0 D4 C8 C5 AB
2568: A0 D2 C1 CE C7 C5 A0 B1 25
2570: AD B7 8D D7 C9 CC CC A0 6A
2578: C2 C5 A0 D4 C1 CB C5 CE 8E
2580: A0 D4 CF A0 CD C5 C1 CE 2C
2588: A0 D4 C8 C1 D4 8D CE CF D7
2590: A0 D0 D2 C9 CE D4 C5 D2 7E
2598: A0 C9 D3 A0 C3 CF CE CE F3
25A0: C5 C3 D4 C5 C4 AA 04 D2 61
25A8: C9 D6 C5 A0 A0 A0 A0 A0 B9
25B0: A0 A0 A0 A0 A0 A0 A0 A0 FA
25B8: A0 A0 A0 A0 A0 A0 A0 A0 03
25C0: A0 A0 A0 A0 A0 A0 12 C5 13
25C8: C1 C4 8D 14 D2 C1 C3 CB 09
25D0: A0 A0 A0 A0 A0 A0 A0 A0 1B
25D8: A0 A0 A0 A0 A0 A0 A0 A0 23

```

```

25E0: A0 A0 A0 A0 A0 A0 A0 A0 2B
25E8: A0 A0 A0 17 D2 C9 D4 C5 5E
25F0: 8D 13 C5 C3 D4 CF D2 A0 E7
25F8: A0 A0 A0 A0 A0 A0 A0 A0 43
2600: A0 A0 A0 A0 A0 A0 A0 A0 4C
2608: A0 A0 A0 A0 A0 A0 A0 A0 54
2610: A0 05 C4 C9 D4 8D DE D3 91
2618: A0 D3 CC CF D4 A0 A0 A0 4B
2620: A0 A0 A0 A0 A0 A0 A0 A0 6C
2628: A0 A0 A0 A0 A0 A0 A0 A0 74
2630: A0 A0 A0 A0 A0 DE D0 A0 D5
2638: D0 D2 C9 CE D4 8D DE D4 37
2640: A0 D4 CF C7 C7 CC C5 A0 26
2648: A0 A0 A0 A0 A0 A0 A0 A0 94
2650: A0 A0 A0 A0 A0 A0 A0 A0 9C
2658: A0 A0 A0 A0 A0 DE D8 A0 0E
2660: C5 D8 C9 D4 AA C1 D3 C3 14
2668: C9 C9 A0 CD D3 C2 A0 CF B7
2670: CE AA C1 D3 C3 C9 C9 A0 BD
2678: CD D3 C2 A0 CF C6 C6 AA D4
2680: C8 C5 D8 A0 A0 A0 A0 A0 31
2688: A0 A0 A0 A0 A0 AA D0 D2 8F
2690: C5 D3 D3 A0 DE DA A0 D4 B1
2698: CF A0 D4 C5 D2 CD C9 CE 1C
26A0: C1 D4 C5 A0 C5 C4 C9 D4 6F
26A8: A0 CD CF C4 C5 AA C1 D2 2E
26B0: C5 A0 D9 CF D5 A0 D3 D5 EE
26B8: D2 C5 A0 D9 CF D5 A0 D7 80
26C0: C1 CE D4 A0 D4 CF A0 D7 45
26C8: D2 C9 D4 C5 A0 A8 D9 AF F2
26D0: CE A9 BF AA D4 D2 C1 C3 CA
26D8: CB BA A0 A0 A0 A0 A0 A0 41
26E0: A0 D3 C5 C3 D4 CF D2 BA AD
26E8: A0 A0 A0 8D 8D AA C5 D2 10
26F0: D2 CF D2 A0 A0 A0 AD AD 8F
26F8: A0 A0 A0 D0 D2 C5 D3 D3 08
2700: A0 C1 CE D9 A0 CB C5 D9 20
2708: A0 D4 CF A0 C3 CF CE D4 AF
2710: C9 CE D5 C5 A0 A0 A0 A0 77
2718: A0 B0 A0 A0 A0 B0 A0 A0 AA
2720: A0 B1 A0 80 A0 FF 00 00 4C

```

Program 2. Dr. Disk, ProDOS

For mistake-proof program entry, use "AppleMLX" (Appendix C) to type in this program.

START ADDRESS: 2000
END ADDRESS: 272F

```

2000: DB 20 58 FC A9 4D 85 F9 17
2008: A9 25 85 FA 20 5E 24 20 A9
2010: F0 24 C9 B1 90 09 C9 B8 1B
2018: B0 05 29 07 4C 21 20 A9 58
2020: 80 8D 2B 27 38 6E 2C 27 D6
2028: A9 06 8D 22 27 A9 01 8D 02
2030: 21 27 A9 00 8D 10 22 8D 7E
2038: 11 22 AD 21 27 0A 0A 29 EF
2040: 08 0D 22 27 0A 0A 0A 0A 15
2048: 8D 0D 22 A9 00 8D 0E 22 E5
2050: 85 FB 8D 1F 27 8D 20 27 CC
2058: 8D 24 27 8D 26 27 A9 03 4A
2060: 8D 0C 22 A9 10 8D 0F 22 40
2068: 85 FC A9 FF 8D 25 27 20 4F
2070: 58 FC A9 CA 85 F9 A9 25 8A
2078: 85 FA A9 11 85 25 A9 00 94
2080: 85 24 20 66 24 2C 2C 27 48
2088: 30 03 4C 27 24 2C 26 27 E2
2090: 10 03 4C A9 22 A9 12 85 1F
2098: 25 20 BD 24 A0 08 84 24 BF
20A0: 88 AD 21 27 09 B0 91 28 7D
20A8: A9 11 85 25 20 BD 24 AD F2
20B0: 22 27 09 B0 91 28 A9 13 8B
20B8: 85 25 20 BD 24 AD 11 22 01
20C0: 20 6A 23 91 28 AD 10 22 63
20C8: 20 DA FD 2C 24 27 30 1B 8B
20D0: 2C 25 27 30 0B A9 82 85 E1
20D8: F9 A9 26 85 FA 4C F3 20 AE
20E0: A9 75 85 F9 A9 26 85 FA 8F
20E8: 4C F3 20 A9 90 85 F9 A9 23
20F0: 26 85 FA A9 18 85 24 A9 68
20F8: 14 85 25 20 66 24 A9 00 62
2100: 85 24 A9 17 85 25 20 BD 73
2108: 24 20 9C FC 20 E5 23 20 C6
2110: DE 24 A9 00 8D 23 27 20 67
2118: E6 24 A0 00 D9 3A 25 F0 DD
2120: 09 C8 BE 3A 25 D0 F5 4C 39
2128: 17 21 98 48 A4 24 88 A9 46
2130: A0 91 28 68 0A A8 C8 B9 F0
2138: 16 25 48 88 B9 16 25 48 19
2140: 60 A9 08 85 24 A9 11 85 E5
2148: 25 20 DB 24 20 FC 24 C9 E9
2150: 08 B0 0B 2C 23 27 30 06 03
2158: 8D 22 27 4C 7D 21 4C 8D 2A

```


2160: 20 A9 08 85 24 A9 12 85 E7
2168: 25 20 DB 24 20 FC 24 F0 31
2170: 1D C9 03 B0 19 2C 23 27 06
2178: 30 14 8D 21 27 AD 21 27 F4
2180: 0A 0A 29 08 0D 22 27 0A 39
2188: 0A 0A 0A 8D 0D 22 4C 8D 83
2190: 20 A9 08 85 24 A9 13 85 1A
2198: 25 20 DB 24 20 FC 24 2C 9C
21A0: 23 27 30 EA 8D 1E 27 20 46
21A8: C5 24 2C 23 27 30 DF 8D D5
21B0: 10 22 AD 1E 27 8D 11 22 CE
21B8: 4C 8D 20 A9 80 8D 15 22 A9
21C0: 18 6E 2C 27 20 12 22 AD DD
21C8: 0F 22 85 FC A9 00 8D 1F 23
21D0: 27 8D 20 27 4C 6F 20 20 01
21D8: 80 FE A9 BE 85 F9 A9 26 C9
21E0: 85 FA A9 00 85 24 A9 17 01
21E8: 85 25 20 66 24 20 84 FE 4B
21F0: A0 26 A9 60 91 28 20 E6 9C
21F8: 24 C9 D9 D0 0C A9 81 8D 9F
2200: 15 22 20 12 22 18 6E 2C F6
2208: 27 4C 8D 20 03 A0 00 10 51
2210: A0 A0 20 00 BF 00 0C 22 09
2218: 90 1F 6E 2C 27 A9 F4 85 4C
2220: F9 A9 26 85 FA A9 00 85 EC
2228: 24 A9 17 85 25 20 80 FE CD
2230: 20 66 24 20 84 FE 20 F0 F5
2238: 24 60 AD 1F 27 38 E9 08 44
2240: 8D 1F 27 09 87 4C 50 22 B8
2248: CE 1F 27 AD 1F 27 09 80 A3
2250: 49 FF D0 2C AD 1F 27 29 77
2258: 80 D0 28 4C 6F 20 AD 1F 51
2260: 27 18 69 08 8D 1F 27 29 4C
2268: 78 4C 72 22 EE 1F 27 AD 5C
2270: 1F 27 29 7F D0 0A AD 1F 54
2278: 27 29 80 F0 06 4C 6F 20 1A
2280: 4C 8D 20 AD 20 27 49 01 5E
2288: 8D 20 27 A5 FC 49 01 85 6F
2290: FC 4C 6F 20 AD 24 27 49 EB
2298: 80 8D 24 27 30 08 AD 25 99
22A0: 27 49 80 8D 25 27 4C 8D 9F
22A8: 20 A9 80 8D 26 27 20 E5 44
22B0: 23 20 DE 24 A9 00 85 24 29
22B8: A9 17 85 25 A9 9E 85 F9 67
22C0: A9 26 85 FA 20 66 24 2C D2
22C8: 24 27 30 12 A9 00 8D 23 9B
22D0: 27 20 F0 24 C9 9A F0 22 CD
22D8: 2D 25 27 4C E1 22 20 C5 44
22E0: 24 48 A9 A0 A4 24 88 91 E0
22E8: 28 68 2C 23 27 30 0B AC CF

```

22F0: 1F 27 91 FB 20 0A 24 4C 3E
22F8: 6C 22 A9 00 8D 26 27 4C D0
2300: 8D 20 2C 2B 27 10 03 4C 19
2308: 8D 20 A0 03 B9 36 00 99 A1
2310: 27 27 8B 10 F7 A9 00 85 B1
2318: 36 AD 2B 27 09 C0 85 37 4A
2320: A9 1B 85 38 A9 FD 85 39 BF
2328: 20 04 24 AD 0E 22 85 FD E0
2330: AD 0F 22 85 FE A9 00 85 D1
2338: F7 85 FB AD 11 22 20 6A 91
2340: 23 8D EF 26 AD 10 22 29 F6
2348: F0 6A 6A 6A 6A 20 6A 23 61
2350: 8D F0 26 AD 10 22 20 6A EC
2358: 23 8D F1 26 A9 E4 85 F9 1A
2360: A9 26 85 FA 20 66 24 4C 94
2368: 79 23 29 0F C9 0A 30 06 27
2370: 38 E9 09 09 C0 60 09 B0 49
2378: 60 A5 F8 C9 02 F0 54 20 B0
2380: DA FD A5 F7 20 DA FD A9 F9
2388: A0 20 ED FD 20 ED FD A0 1A
2390: 00 B1 FD 20 DA FD A9 A0 C7
2398: 20 ED FD C8 C0 10 D0 F1 90
23A0: 20 ED FD A0 00 B1 FD 29 28
23A8: 7F C9 20 B0 02 A9 AE 20 64
23B0: ED FD C8 C0 10 D0 EE 20 54
23B8: 04 24 18 A5 FD 69 10 85 A2
23C0: FD 90 02 E6 FE 18 A5 F7 74
23C8: 69 10 85 F7 90 AB E6 F8 F1
23D0: 4C 79 23 A0 03 B9 27 27 7E
23D8: 99 36 00 8B 10 F7 4C 6F 6A
23E0: 20 20 58 FC 60 AD 1F 27 39
23E8: 29 07 85 F9 0A 65 F9 69 19
23F0: 08 85 24 AD 1F 27 29 78 5C
23F8: 4A 4A 4A 18 69 00 85 25 3D
2400: 20 BD 24 60 A9 8D 20 ED 04
2408: FD 60 20 E5 23 A4 24 88 46
2410: A9 A0 91 28 AC 1F 27 B1 EB
2418: FB 48 20 DA FD 18 A5 F9 B7
2420: 69 20 A8 68 91 28 60 AD 5C
2428: 1F 27 48 29 80 8D 1F 27 05
2430: 20 E5 23 A9 00 85 24 AD 0D
2438: 20 27 20 DA FD AD 1F 27 18
2440: 20 DA FD 20 0A 24 EE 1F EE
2448: 27 AD 1F 27 29 07 D0 F3 E0
2450: AD 1F 27 29 7F D0 D9 68 0A
2458: 8D 1F 27 4C 8D 20 A9 80 99
2460: 8D 23 27 4C 6B 24 A9 00 21
2468: 8D 23 27 20 BD 24 A0 00 E6
2470: B1 F9 8D 1C 27 2C 23 27 DA
2478: F0 2A 8C 1D 27 A9 14 85 B4

```

2480: 24 A9 00 8D 1E 27 AD 1E 25
 2488: 27 49 80 8D 1E 27 30 02 8F
 2490: C6 24 C8 B1 F9 CD 1C 27 DF
 2498: F0 07 C9 8D F0 03 4C 86 DF
 24A0: 24 AC 1D 27 20 AD 24 CD 0A
 24A8: 1C 27 D0 C9 60 C8 B1 F9 03
 24B0: CD 1C 27 F0 07 20 ED FD 6D
 24B8: C9 8D D0 F1 60 48 A5 25 17
 24C0: 20 C1 FB 68 60 20 FC 24 31
 24C8: 2C 23 27 30 0D 0A 0A 0A 86
 24D0: 0A 8D 1D 27 20 00 25 0D EF
 24D8: 1D 27 60 20 BD 24 A4 24 73
 24E0: 88 A9 60 91 28 60 20 F0 F0
 24E8: 24 C9 E0 90 02 29 DF 60 AF
 24F0: 2C 00 C0 10 FB AD 00 C0 BF
 24F8: 2C 10 C0 60 18 6E 23 27 61
 2500: 20 F0 24 49 B0 C9 0A 90 01
 2508: 0A 69 88 C9 FA B0 04 38 3A
 2510: 6E 23 27 29 0F 60 60 21 AD
 2518: 40 21 90 21 BA 21 D6 21 18
 2520: 93 22 A8 22 39 22 39 22 DA
 2528: 47 22 47 22 6B 22 6B 22 86
 2530: 5D 22 5D 22 5D 22 01 23 18
 2538: E0 23 C4 D3 C2 D2 D7 D4 77
 2540: C5 C9 8B CA 88 CB 95 CD 6A
 2548: 8A 8D 90 98 00 AA D7 C8 F9
 2550: C1 D4 A0 D3 CC CF D4 A0 F1
 2558: C9 D3 A0 D9 CF D5 D2 A0 4A
 2560: D0 D2 C9 CE D4 C5 D2 8D DE
 2568: C3 CF CE CE C5 C3 D4 C5 FB
 2570: C4 A0 D4 CF BF 8D 8D C1 ED
 2578: A0 D2 C5 D3 D0 CF CE D3 F4
 2580: C5 A0 CF D5 D4 D3 C9 C4 7B
 2588: C5 A0 D4 C8 C5 A0 D2 C1 1D
 2590: CE C7 C5 A0 B1 AD B7 8D 38
 2598: D7 C9 CC CC A0 C2 C5 A0 E3
 25A0: D4 C1 CB C5 CE A0 D4 CF 0E
 25A8: A0 CD C5 C1 CE A0 D4 C8 F6
 25B0: C1 D4 8D CE CF A0 D0 D2 25
 25B8: C9 CE D4 C5 D2 A0 C9 D3 13
 25C0: A0 C3 CF CE CE C5 C3 D4 1D
 25C8: C5 C4 AA 13 CC CF D4 A0 9D
 25D0: A0 A0 A0 A0 A0 A0 A0 A0 1B
 25D8: A0 A0 A0 A0 A0 A0 A0 A0 23
 25E0: A0 A0 A0 12 C5 C1 C4 A0 38
 25E8: A0 C2 CC CF C3 CB 8D 04 37
 25F0: D2 C9 D6 C5 A0 A0 A0 A0 B7
 25F8: A0 A0 A0 A0 A0 A0 A0 A0 43
 2600: A0 A0 A0 A0 A0 A0 A0 17 C2
 2608: D2 C9 D4 C5 A0 C2 CC CF A0

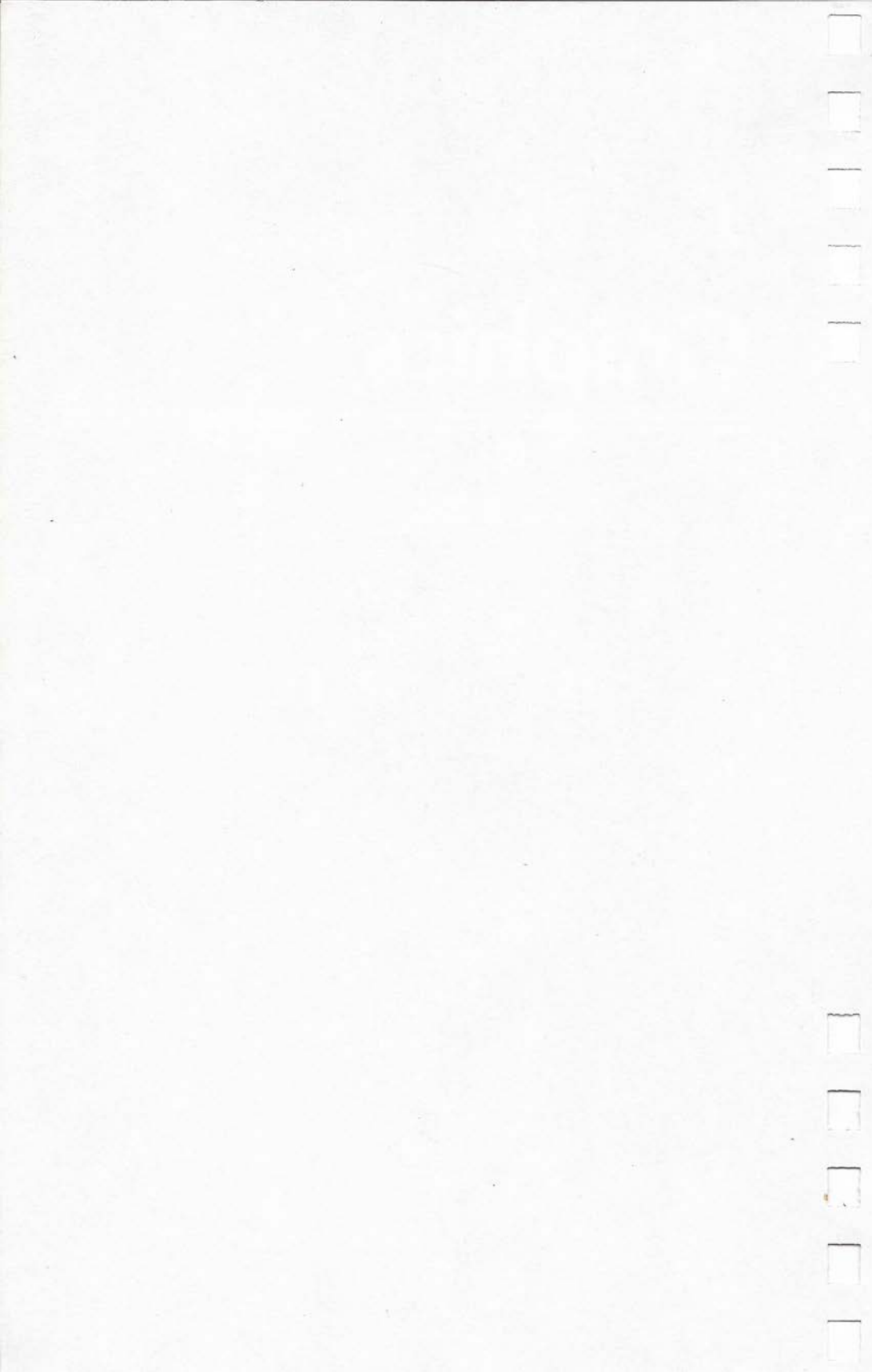

```

2610: C3 CB 8D 02 CC CF C3 CB FB
2618: A0 A0 A0 A0 A0 A0 A0 A0 64
2620: A0 A0 A0 A0 A0 A0 A0 A0 6C
2628: A0 A0 A0 05 C4 C9 D4 A0 E8
2630: A0 C2 CC CF C3 CB 8D 14 90
2638: CF C7 C7 CC C5 A0 C1 D3 2C
2640: C3 C9 C9 A0 BC AD AD BE DA
2648: A0 C8 C5 D8 8D DE D0 A0 87
2650: D0 D2 C9 CE D4 A0 C2 CC 5B
2658: CF C3 CB A0 A0 A0 A0 A0 6A
2660: A0 A0 A0 A0 A0 DE D8 A0 16
2668: C5 D8 C9 D4 A0 A0 A0 A0 8D
2670: A0 A0 A0 A0 A0 AA C1 D3 5A
2678: C3 C9 C9 A0 CD D3 C2 A0 40
2680: CF CE AA C1 D3 C3 C9 C9 E4
2688: A0 CD D3 C2 A0 CF C6 C6 D7
2690: AA C8 C5 D8 A0 A0 A0 A0 14
2698: A0 A0 A0 A0 A0 A0 AA D0 29
26A0: D2 C5 D3 D3 A0 DE DA A0 56
26A8: D4 CF A0 D4 C5 D2 CD C9 93
26B0: CE C1 D4 C5 A0 C5 C4 C9 3B
26B8: D4 A0 CD CF C4 C5 AA C1 A2
26C0: D2 C5 A0 D9 CF D5 A0 D3 84
26C8: D5 D2 C5 A0 D9 CF D5 A0 D1
26D0: D7 C1 CE D4 A0 D4 CF A0 39
26D8: D7 D2 C9 D4 C5 A0 A8 D9 28
26E0: AF CE A9 BF AA 8D 8D C2 53
26E8: CC CF C3 CB BA A0 A0 C1 20
26F0: B7 A0 8D 8D AA C5 D2 D2 B0
26F8: CF D2 A0 A0 A0 AD AD A0 B7
2700: A0 A0 D0 D2 C5 D3 D3 A0 D3
2708: C1 CE D9 A0 CB C5 D9 A0 F9
2710: D4 CF A0 C3 CF CE D4 C9 3A
2718: CE D5 C5 A0 A0 A0 B1 A0 91
2720: A0 A0 A0 C4 A0 80 A0 A0 30
2728: FF A0 C3 A0 C9 00 00 00 6F

```


4

Graphics



Apple Animator

Steve Johnson

Apple version by Tim Victor

This feature-packed utility makes it a breeze to create your own short cartoons or animation sequences on any Apple II series computer screen, using either DOS 3.3 or ProDOS.

Computer animation can be marvelous to behold, but a drudge to produce. Whether you're working in BASIC or machine language, creating objects and manipulating them on the screen can mean fumbling for hours with PEEKs, POKEs, bits, bytes, and other tedious details.

"Apple Animator" goes a long way toward automating this process. It works much like a cartoonist's sketch pad, letting you draw a series of similar images which are then displayed in rapid sequence to create the illusion of movement. Your finished cartoons can even be saved on disk and reloaded for viewing later.

Apple Animator requires two files on disk: the main BASIC program and a binary file (ANIMATOR2) that contains graphics data. Program 1 is the graphics data, which must be entered using the "Apple MLX" machine language editor from Appendix C. Be sure you read and understand the instructions for using MLX before you begin entering the data. Before you load and run MLX (or before you reload MLX if you enter the data in more than one sitting), you should enter the following line in direct mode (without a line number) and press RETURN:

HIMEM: 32256

Then load and run MLX. You'll be asked for a starting address and an ending address. For the Animator graphics data file, the proper values are as follows:

Starting address: 7E00

Ending address: 83AF

When you finish entering the data, be sure to save a copy before you leave MLX. For Apple Animator to work correctly, you must use the filename ANIMATOR2 for the graphics file (see line 100 of Program 2). Next, type in and save Program 2, the actual Apple Animator program, and save it on the same disk with the ANIMATOR2 graphics file. To start the program, simply load and run Program 2.

Drawing an Image

When you run Apple Animator, it displays an editing screen with 20 numbered frames. You can draw as many as 20 pictures, one in each frame, then flip rapidly through the frames to create animation. The frame number displayed at the upper left of the screen shows which frame you're currently working on. Normally, Apple Animator begins the animation with frame 1 and ends with frame 20. But you can start and end the animation wherever you like. For example, a short sequence might start with frame 1 and end with frame 3. To view only part of a long sequence, you might start at frame 12 and end at frame 18. The frame number is selected by pressing the right- and left-arrow keys.

The frame number also determines which frame you'll be working on when you go to the editing screen. Let's start with a simple example. Make sure the frame number is set to 1; then press 0 to select the editing function and press Return at the next prompt. After a brief pause, Apple Animator displays a drawing grid with a blinking cursor. Edit mode has three main functions and several secondary features. Each one is chosen by pressing a different key.

Key	Function
-----	----------

- | | |
|---|--------|
| 1 | Quit |
| 2 | Draw |
| 3 | Move |
| 4 | Erase |
| 5 | Clear |
| 6 | Save |
| 7 | Invert |
| 8 | Update |
| 9 | Revert |

The functions you'll use most often are Draw (2), Move (3), and Erase (4). Notice that when you first enter the editing screen Move is already selected, as indicated by the bullet (•) beside it.

For now, concentrate on just these three features. *Draw* draws wherever the cursor moves, *Erase* erases wherever the cursor happens to be, and *Move* lets you move the cursor without disturbing anything on the screen.

When the editing screen appears, move the blinking cursor (which first shows in the upper-left corner of the grid) left, right, up, and down by pressing the J, L, I, and K keys, respectively.

Draw a simple shape on the grid to become familiar with these basic functions. Though Apple Animator does not display the shape in actual size while you draw, you can see it displayed by pressing the 8 (Update) key. The shape appears on the right. An invert function (7) lets you reverse everything on the grid—every dot becomes a blank, and vice versa. Be patient—it takes Apple Animator a while to complete this process.

Once the picture is finished, you can press 6 to save it and return to the main screen. *Note that you must save a picture by pressing 6 (Save) to put it into the frame on the main screen.* If you exit the edit mode by pressing 1 (Quit), the new picture is lost, and Apple Animator uses whatever that frame previously contained. Try drawing a simple shape and saving it. Since this is just for practice, any scribble will do. When you return to the main screen, Apple Animator displays the picture in frame 1.

Frame by Frame

Now you're ready to draw the next frame in the sequence. In most cases, you'll want to make only slight changes from one frame to the next to simulate smooth motion. To save time, Apple Animator lets you copy a picture from one frame to another. Let's demonstrate this by copying the picture from frame 1 to frame 2. Set the picture number to 1 with the arrow keys, then press 0 to edit. Apple Animator displays a prompt, inviting you to enter a frame number. To edit the current picture number, you would just press Return. However, by entering a *different* number you can copy the current picture into a different frame, then change that picture to make the next frame in your cartoon.

When you enter 2 (for frame 2) at the prompt, Animator copies the picture from frame 1 into the drawing grid. When the drawing grid appears, make some change in the picture to distinguish it from frame 1. Now press 6 to save the picture in frame 2 and return to the main screen. Apple Animator displays both pictures in their respective frames.

After drawing a few frames, you're ready to bring them to life. The first step is to specify the starting and ending frame numbers. The starting number determines which frame begins the animation, and the ending number tells Apple Animator where the series ends.

Set the starting number first. Use the arrow keys to set the frame number to 1, then press the 3 key. Now use the arrow keys to make the frame number match the *last* frame that contains a picture—press the 4 key. This sets the ending number. You must always set the starting and ending numbers before selecting animation (if you don't, Apple Animator flips through all 20 frames whether they contain pictures or not). Once these numbers are set, press 5 to view the animation sequence. Hit the space bar to pause and Return to stop.

By selecting different *speed* and *pause* values, you can move the animated figure across the screen. The speed value can range from -15 to 15. When it's 0, the figure is animated in place. Positive values move the figure from left to right; negative values move it from right to left. The greater the value, the faster the figure moves. Press 6 to increase the animation speed and 7 to decrease it.

The pause value controls the time delay between each frame of the animation. A small pause value makes the pictures change very quickly, while larger values slow down the process. Pressing 8 selects a smaller pause; 9 selects a larger value.

Macro Editing Features

Apple Animator provides a few macro (large-scale) editing features to help you work with longer cartoons. All these features are accessible only from the main screen. The insert function lets you insert a blank frame anywhere in the series. To use it, set the frame number to the number of the frame where you want to insert a blank, then press 1. The designated picture and all those following it are bumped forward one frame. When you insert, the picture in frame 20 is always lost.

The delete function lets you delete any frame in the series. Change the picture number to the frame you want to eliminate, then press 2. All the higher numbered pictures move down one frame, deleting the picture in the designated frame. Frame 20 is always blank after a deletion.

The invert function (press D) works just like invert in editing mode, but it inverts all 20 frames at once.

To clear all 20 frames, press E to quit or C to clear. Since these last two functions can have drastic results, Animator lets you abort either one without causing harm.

When you finish a sequence, press B to save it on disk.

The screen clears and displays two options: You can press Esc to cancel the save or Return to list the picture files on that disk. If you simply want to save the animation sequence, type in a picture filename. Since Apple Animator doesn't add a file-name suffix for you, it's a good idea to append *.ANI* to the end of filenames yourself. This makes it much easier to tell which files on a disk are picture files.

Other Editing Features

Besides the primary editing functions (Draw, Move, and Erase), Apple Animator provides a number of other useful features. *Clear*, selected by pressing 5, clears the screen, giving you an empty frame. Additional choices, however, are offered in a menu near the bottom of the screen. If you just want the frame to remain black, press 0 and Return. Notice, though, that other colors are available, from Green (1) to White2 (7). Clearing the frame and choosing a color creates the pixel combinations which represent that color. You can draw on it as usual, and, after saving the frame, it will appear in the selected color on the main screen. (Of course, you'll be able to see this only if you have a color monitor or television connected to your Apple.)

As mentioned earlier, choose *Update* by hitting the 8 key. Almost immediately you'll see an actual-scale reproduction of the shape displayed on the right of the screen. You can use this feature to check the shape from time to time as you're drawing or editing.

Revert, selected with the 9 key, returns the shape to what it was the last time it was saved from the editing screen (not from the disk file). This can come in handy when you've been editing a frame, but decide you like it better the way it was before.

Program 1. ANIMATOR2 Graphics File

For mistake-proof program entry, use the "Apple MLX" (Appendix C) to type in this program.

START ADDRESS: 7E00
END ADDRESS: 83AF

```
7E00: 00 00 00 00 00 00 00 00 FC
7E08: 00 00 00 00 00 00 00 00 05
7E10: D8 78 85 45 86 46 84 47 3A
7E18: A6 07 0A 0A B0 04 10 3E FF
```



```

7E20: 30 04 10 01 E8 E8 0A 86 CD
7E28: 1B 18 65 06 85 1A 90 02 7D
7E30: E6 1B A5 28 85 08 A5 29 5F
7E38: 29 03 05 E6 85 09 A2 08 37
7E40: A0 00 B1 1A 24 32 30 02 B1
7E48: 49 7F A4 24 91 08 E6 1A 35
7E50: D0 02 E6 1B A5 09 18 69 AF
7E58: 04 85 09 CA D0 E2 A5 45 29
7E60: A6 46 A4 47 58 4C F0 FD 1F
7E68: 80 80 80 80 BE 80 80 80 57
7E70: 80 80 80 BE 80 BE 80 80 4A
7E78: 80 BC E6 B0 98 80 98 80 45
7E80: 80 BC E6 F6 EE E6 BC 80 46
7E88: 80 98 9C 98 98 98 BC 80 2A
7E90: 80 BC E6 B0 8C E6 FE 80 63
7E98: 80 BC E6 B0 E0 E6 BC 80 89
7EA0: 80 B0 B8 B4 FE B0 B0 80 09
7EA8: 80 FE 86 BE E0 E6 BC 80 FE
7EB0: 80 BC 86 BE E6 E6 BC 80 A6
7EB8: 80 FE E0 B0 98 8C 8C 80 6D
7EC0: 80 BC E6 BC E6 E6 BC 80 A2
7EC8: 80 BC E6 E6 FC B0 98 80 DC
7ED0: 80 98 B0 FE FE B0 98 80 A6
7ED8: 80 BE BE BE BE BE BE BE 80 78
7EE0: 00 00 00 00 00 00 00 00 DD
7EE8: 00 00 00 00 00 00 00 00 E5
7EF0: 00 00 00 00 00 00 00 00 ED
7EF8: 00 00 00 00 00 00 00 00 F5
7F00: 80 80 98 BC BC 98 80 80 08
7F08: 80 FC E6 E6 FE E6 E6 80 B3
7F10: 80 BE E6 E6 BE E6 FE 80 5A
7F18: 80 BC E6 86 86 E6 BE 80 99
7F20: 80 BE E6 E6 E6 E6 BE 80 2B
7F28: 80 FE 86 86 BE 86 FE 80 EE
7F30: 80 FE 86 86 BE 86 86 80 06
7F38: 80 BC E6 86 F6 E6 BE 80 3D
7F40: 80 E6 E6 E6 FE E6 E6 80 66
7F48: 80 98 98 98 98 98 98 80 23
7F50: 80 E0 E0 E0 E0 E6 BC 80 8E
7F58: 80 E6 E6 B6 9E E6 E6 80 78
7F60: 80 86 86 86 86 86 FE 80 47
7F68: 80 E6 FE E6 E6 E6 E6 80 D0
7F70: 80 BE E6 E6 E6 E6 E6 80 CB
7F78: 80 BC E6 E6 E6 E6 BC 80 FE
7F80: 80 BE E6 E6 BE 86 86 80 58
7F88: 80 BC E6 E6 E6 B6 EC 80 AE
7F90: 80 BE E6 E6 BE E6 E6 80 AA
7F98: 80 BC E6 8C B0 E6 BE 80 CB
7FA0: 80 FE 98 98 98 98 98 80 15
7FAB: 80 E6 E6 E6 E6 E6 BE 80 BD

```

```

7FB0: 80 E6 E6 E6 E6 E6 98 80 79
7FB8: 80 E6 E6 E6 E6 FE E6 80 7E
7FC0: 80 E6 E6 E6 BC E6 E6 80 D4
7FC8: 80 E6 E6 E6 BC 98 98 80 07
7FD0: 80 FE B0 98 8C 86 FE 80 6C
7FD8: 00 00 00 00 00 00 00 00 D7
7FE0: A0 00 A2 07 A9 C4 85 FE 5F
7FE8: A9 84 85 FF B1 FE 49 7F 2A
7FF0: 91 FE C8 D0 F7 E6 FF CA C4
7FF8: D0 F2 60 00 00 00 00 00 29
8000: 4C 15 80 4C 55 80 4C 82 09
8008: 80 4C EF 80 4C 06 82 4C 2E
8010: 94 81 4C D2 81 A9 02 8D B6
8018: B4 83 A9 18 8D B5 83 20 75
8020: 1F 82 B0 30 20 47 83 B0 40
8028: 2B 20 9C 83 B0 26 AD B2 BE
8030: 83 85 FC AD B3 83 85 FD 83
8038: 20 DF 82 20 5F 82 20 A8 81
8040: 82 EE B8 83 A5 FC 18 69 48
8048: 03 85 FC 90 02 E6 FD CE 4B
8050: B5 83 D0 E4 60 A9 02 8D B0
8058: B4 83 A9 18 8D B5 83 20 B5
8060: 47 83 B0 1D 20 9C 83 B0 FB
8068: 18 A0 03 A9 00 99 C4 83 0C
8070: 88 10 FA 20 DF 82 20 A8 0D
8078: 82 EE B8 83 CE B5 83 D0 EA
8080: F2 60 A5 38 A4 39 C9 BA A3
8088: D0 04 C0 80 F0 0E 8D BE AC
8090: 83 8C BF 83 A9 BA 85 38 22
8098: A0 80 84 39 20 79 81 A9 C1
80A0: 00 8D C2 83 20 1F 82 B0 C8
80A8: 0D AD B2 83 8D C5 80 AD 5C
80B0: B3 83 8D C6 80 60 20 06 56
80B8: 82 00 91 28 A9 30 8D 00 D8
80C0: 02 8D 01 02 AD FF FF C9 9D
80C8: 64 90 07 E9 64 EE 00 02 80
80D0: D0 F5 C9 0A 90 07 E9 0A 10
80D8: EE 01 02 D0 F5 69 30 8D 22
80E0: 02 02 A2 03 A9 8D EE C5 0F
80E8: 80 D0 03 EE C6 80 60 20 C6
80F0: 79 81 A9 FF 8D C2 83 20 E2
80F8: 1F 82 B0 17 AD B2 83 8D 7E
8100: 54 81 AD B3 83 8D 55 81 FC
8108: A9 C4 8D 6E 81 A9 83 8D F0
8110: 6F 81 60 20 06 82 00 2C 9F
8118: C2 83 30 03 4C F0 FD C9 7F
8120: B0 90 04 C9 BA 90 46 48 A9
8128: 8C C3 83 38 AD 6E 81 E9 6A
8130: C4 F0 35 8D B6 83 A9 00 68
8138: A8 C9 1A B0 D6 0A 8D 44 8E

```

```

8140: 81 0A 0A 69 FF B0 CC 79 34
8148: C4 83 38 E9 B0 C8 CC B6 2D
8150: 83 D0 E6 8D FF FF EE 54 31
8158: 81 D0 03 EE 55 81 A9 C4 68
8160: 8D 6E 81 A9 83 8D 6F 81 43
8168: AC C3 83 68 60 8D FF FF E2
8170: EE 6E 81 D0 03 EE 6F 81 F7
8178: 60 A5 36 A4 37 C9 17 D0 06
8180: 04 C0 81 F0 0E 8D C0 83 A0
8188: 8C C1 83 A9 17 85 36 A0 29
8190: 81 84 37 60 20 1F 82 B0 95
8198: 38 AD B2 83 8D C6 81 D0 0D
81A0: 03 CE B3 83 CE C6 81 AD C9
81A8: B3 83 8D CC 81 A9 E3 85 E4
81B0: FC A9 8E 85 FD A0 00 B1 EA
81B8: FC A0 48 91 FC A5 FC D0 CD
81C0: 02 C6 FD C6 FC A9 FF C5 F6
81C8: FC D0 EA A9 FF C5 FD D0 5A
81D0: E4 60 20 1F 82 B0 2E AD 35
81D8: B2 83 85 FC AD B3 83 85 5F
81E0: FD A9 E4 8D FA 81 A9 8E 82
81E8: 8D 00 82 A0 48 B1 FC A0 B0
81F0: 00 91 FC E6 FC D0 02 E6 7C
81F8: FD A9 FF C5 FC D0 EC A9 70
8200: FF C5 FD D0 E6 60 AD BE 16
8208: 83 AC BF 83 F0 04 85 38 05
8210: 84 39 AD C0 83 AC C1 83 3D
8218: F0 04 85 36 84 37 60 A9 16
8220: C4 8D B2 83 A9 84 8D B3 A7
8228: 83 20 A5 83 C9 15 90 01 A8
8230: 60 8D B0 83 A9 00 8D B1 31
8238: 83 A0 03 20 40 82 A0 03 D9
8240: AD B0 83 0A 2E B1 83 88 21
8248: D0 F9 8D B0 83 18 6D B2 FA
8250: 83 8D B2 83 AD B3 83 6D B9
8258: B1 83 8D B3 83 18 60 AC ED
8260: B4 83 C8 8C B6 83 A9 00 99
8268: 99 C4 83 88 B1 FC 99 C4 DD
8270: 83 88 10 F8 AD C4 83 09 7B
8278: 7F 8D B7 83 AC BB 83 F0 1C
8280: 15 A2 00 0E C4 83 BD C4 0E
8288: 83 0A 3E C5 83 E8 EC B6 46
8290: 83 D0 F3 88 D0 EB AC B6 D8
8298: 83 B9 C4 83 09 80 2D B7 FA
82A0: 83 99 C4 83 88 10 F2 60 69
82A8: AC BB 83 B9 D1 82 AC B6 A7
82B0: 83 31 FE 19 C4 83 91 FE 8B
82B8: 88 B9 C4 83 91 FE 88 D0 AB
82C0: F8 AC BB 83 B9 D8 82 A0 F3
82C8: 00 31 FE 0D C4 83 91 FE 21

```



```

82D0: 60 7F 7E 7C 78 70 60 40 04
82D8: 00 01 03 07 0F 1F 3F AD 10
82E0: B8 83 29 3F A8 B9 07 83 F9
82E8: 05 E6 85 FF AD B8 83 29 5B
82F0: 08 F0 02 A9 80 18 2C B8 86
82F8: 83 70 04 10 04 69 28 69 DC
8300: 28 6D BA 83 85 FE 60 00 EE
8308: 04 08 0C 10 14 18 1C 00 CE
8310: 04 08 0C 10 14 18 1C 01 D7
8318: 05 09 0D 11 15 19 1D 01 DE
8320: 05 09 0D 11 15 19 1D 02 E7
8328: 06 0A 0E 12 16 1A 1E 02 EE
8330: 06 0A 0E 12 16 1A 1E 03 F7
8338: 07 0B 0F 13 17 1B 1F 03 FE
8340: 07 0B 0F 13 17 1B 1F A9 AD
8348: 00 8D BA 83 8D B8 83 20 C4
8350: A5 83 8D B9 83 C0 01 90 0A
8358: 12 F0 01 60 C9 18 90 01 9B
8360: 60 A9 24 8D BA 83 A9 04 9A
8368: 8D B8 83 A9 00 8D BC 83 63
8370: A9 E0 8D BD 83 AD B9 83 DB
8378: CD BD 83 90 04 ED BD 83 26
8380: 38 2E BC 83 4E BD 83 90 FF
8388: EF 18 6D 9B 83 8D B8 83 44
8390: 18 AD BC 83 6D BA 83 8D C9
8398: BA 83 18 60 20 A5 83 8D 13
83A0: B8 83 C9 C0 60 20 B1 00 11
83A8: 20 05 E1 A5 A1 A4 A0 60 D8

```

Program 2. Apple Animator

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```

FD 100 DIM EX(2,23),M$(8),CM$(7),MM$(14):D$ = CHR$ (
      4): PRINT D$;"BLOAD ANIMATOR2"
AF 110 FOR I = 0 TO 8: READ M$(I): NEXT I: FOR I = 0
      TO 7: READ CM$(I): NEXT I: FOR I = 0 TO 14: RE
      AD MM$(I): NEXT I
46 120 GOSUB 530
D6 130 A$ = "": FOR I = 1 TO 72: A$ = A$ + "0 ": A = F
      RE (0): NEXT I
16 140 GOSUB 650: GOSUB 590
16 150 F = 1: AB = 1: AE = 20: AS = 0: AP = 10
85 160 ONERR GOTO 1390
6C 170 GOSUB 710: IF MQ = 1 THEN HOME: TEXT: END
44 180 C = PEEK (49152): IF C < 128 THEN 180
45 190 POKE 49168,0: IF C = 136 THEN F = F - 1 + 20
      * (F = 1): GOTO 170

```

```

CF 200 IF C = 149 THEN F = F + 1 - 20 * (F = 20): GO
    TO 170
69 210 C = C - 175: IF C < 1 OR C > 22 THEN 180
CB 220 IF C < 11 THEN 250
75 230 IF C < 18 THEN 180
98 240 C = C - 7
6A 250 MQ = 0: ON C GOSUB 260,1120,1210,1500,1510,15
    20,1660,1670,1680,1690,1370,1250,1470,1460,14
    40:T = FRE (0): GOTO 170
02 260 GOSUB 560: GOSUB 700: VTAB 19: HTAB 14: PRINT
    "EDITING BOX ";A
51 270 PRINT "PRESS ESC TO CANCEL": HTAB 7: PRINT "R
    ETURN FOR SAME": PRINT : PRINT "STORE RESULT
    IN BOX ";R$ = ""
90 280 XC = 21 + LEN (R$): VTAB 23: HTAB XC: PRINT "
    ";
77 290 C = PEEK (49152): IF C < 128 THEN 290
25 300 POKE 49168,0: IF C = 141 OR C = 155 THEN VTAB
    23: HTAB XC: PRINT " "; GOTO 360
3A 310 IF C < > 136 AND C < > 255 THEN 340
AE 320 VTAB 23: HTAB XC: PRINT " "; IF LEN (R$) < 2
    THEN R$ = "": GOTO 280
46 330 R$ = LEFT$ (R$, LEN (R$) - 1): GOTO 280
AD 340 IF C < 176 OR C > 185 THEN 290
93 350 VTAB 23: HTAB XC: PRINT CHR$ (C - 128);R$ =
    R$ + CHR$ (C - 128):Q = FRE (0): GOTO 280
4E 360 IF C = 155 THEN GOSUB 660: RETURN
F6 370 IF R$ = "" THEN AA = A: GOTO 390
06 380 AA = VAL (R$): IF AA > 20 THEN GOSUB 660: RET
    URN
9A 390 HGR2 : HOME : GOSUB 1070
F2 400 XP = 184:YP = 44:DX = 65:DY = 80: GOSUB 980
3A 410 VTAB 7: FOR Q = 0 TO 8: HTAB 28: INVERSE : PR
    INT MID$ ("123456789",Q + 1,1);: NORMAL : PRI
    NT " ";M$(Q): NEXT
BC 420 CALL 32768,A,206,12:XC = 0:YC = 0:QF = 0: GOS
    UB 870
44 430 IF QF THEN GOSUB 590: RETURN
32 440 SC = INT (YC / 8):SA = 1024 + YC * 128 - SC *
    984 + XC:CC$ = " "
84 450 OC$ = CC$:CC$ = CHR$ ( PEEK (SA) - 128): HTAB
    XC + 1: VTAB YC + 1: PRINT OC$;
C4 460 C = PEEK (49152): IF C < 128 THEN T = FRE (0)
    : GOTO 450
2E 470 POKE 49168,0: IF OC$ = " " THEN HTAB XC + 1:
    VTAB YC + 1: PRINT CC$;
91 480 FOR Q = 1 TO 13: IF C < > ASC ( MID$ ("JLIK12
    3456789",Q,1)) + 128 THEN NEXT
B6 490 ON Q GOSUB 750,760,770,780,840,850,870,880,89
    0,990,1020,1050,1060: GOTO 430

```

```

BB 500 I = A * 28 - 24 - 280 * (A > 10): J = 26 + 80
      * (A > 10): CALL 32768, A, I, J
EB 510 X = FRE (0): RETURN
AD 520 FOR A = 1 TO 20: GOSUB 500: NEXT : RETURN
17 530 POKE 6,0: POKE 7,126: IF PEEK (48640) = 76 TH
      EN 550
60 540 POKE 54,16: POKE 55,126: CALL 1002: RETURN
3E 550 PRINT : PRINT CHR$ (4); "PR#A$7E10": RETURN
0B 560 A = F: RETURN
E3 570 GOSUB 580: HTAB 1: INPUT "WHICH BOX?"; A: A = I
      NT (A): IF A < 1 OR A > 20 THEN 570
2F 580 VTAB 22: HTAB 1: PRINT SPC( 39): RETURN
6F 590 HOME : HGR2 : HCOLOR= 3
8A 600 FOR J = 25 TO 105 STEP 80: FOR I = 3 TO I + 9
      * 28 STEP 28: FOR P = 0 TO 1
9D 610 HPlot I - P, J - P TO I + 22 + P, J - P TO I +
      22 + P, J + 25 + P TO I - P, J + 25 + P TO I -
      P, J - P
FB 620 NEXT : NEXT
1B 630 FOR J = 0 TO 1: FOR I = 1 TO 10: HTAB I * 4 -
      2: VTAB J * 10 + 3: PRINT I + J * 10: NEXT :
      NEXT
9D 640 GOSUB 520: GOTO 660
87 650 POKE 242,0: CALL 32777,0: FOR I = 0 TO 20: PR
      INT A$: NEXT : CALL 32780: RETURN
3A 660 GOSUB 700: XP = 2: YP = 140: DX = 275: DY = 48: G
      OSUB 980
AA 670 VTAB 19: FOR I = 0 TO 4: HTAB 2: INVERSE : PR
      INT I;: NORMAL : PRINT " "; MM$(I);: HTAB 17:
      INVERSE : PRINT I + 5;
0B 680 NORMAL : PRINT " "; MM$(I + 5);: HTAB 32: INVE
      RSE : PRINT CHR$ (65 + I);: NORMAL : PRINT "
      "; MM$(I + 10): NEXT
28 690 RETURN
03 700 HTAB 1: VTAB 18: FOR Q = 1 TO 7: PRINT SPC( 4
      0): NEXT : RETURN
07 710 VTAB 1: HTAB 1: PRINT "FRAME "; F; " ";
76 720 HTAB 10: PRINT "RANGE "; AB; " ";: HTAB 18: PRI
      NT " "; AE; " ";
FD 730 HTAB 22: PRINT "SPEED "; AS; " ";: HTAB 32: PR
      INT "PAUSE "; AP; " ";
1F 740 RETURN
1B 750 XC = XC - (XC > 0): GOTO 790
47 760 XC = XC + (XC < 20): GOTO 790
6B 770 YC = YC - (YC > 0): GOTO 790
58 780 YC = YC + (YC < 23): GOTO 790
D1 790 XB = INT (XC / 7): PM = 2 ^ (XC - 7 * XB): T =
      INT (EZ(XB, YC) / PM): ON DF GOTO 800, 820: RET
      URN

```



```

BB 800 IF T = 2 * INT ( T / 2 ) THEN E%(XB,YC) = E%(XB
,YC) + PM: HTAB XC + 1: VTAB YC + 1: PRINT "
";
1A 810 RETURN
5B 820 IF T < > 2 * INT ( T / 2 ) THEN E%(XB,YC) = E%(
XB,YC) - PM: HTAB XC + 1: VTAB YC + 1: PRINT
" - ";
1E 830 RETURN
CB 840 QF = 1: RETURN
5B 850 DF = 1: GOSUB 860: HTAB 29: VTAB 8: PRINT " @ "
;: GOTO 790
85 860 VTAB 8: FOR I = 1 TO 3: HTAB 29: PRINT " " : N
EXT : RETURN
7B 870 DF = 0: GOSUB 860: HTAB 29: VTAB 9: PRINT " @ "
;: RETURN
CD 880 DF = 2: GOSUB 860: HTAB 29: VTAB 10: PRINT " @
";: GOTO 790
D3 890 XP = 149: YP = 140: DX = 128: DY = 40: GOSUB 980
4B 900 FOR I = 0 TO 3: VTAB 19 + I: FOR J = 0 TO 1:
HTAB 23 + 9 * J: INVERSE : PRINT I + J * 4;:
NORMAL : PRINT " "; CM$(I + J * 4);: NEXT : NE
XT
DD 910 C = PEEK (49152): IF C < 128 THEN 910
89 920 VTAB 18: FOR I = 0 TO 5: HTAB 22: PRINT SPC(
19): NEXT : POKE 49168,0:C = C - 176
E1 930 IF C < 0 OR C > 7 THEN RETURN
22 940 T = C - 4 * INT ( C / 4 ): P0 = 42 * T + ( T > 1 )
:P1 = P0: IF P0 = 42 OR P0 = 85 THEN P1 = 127
- P0
8B 950 IF C > 3 THEN P0 = P0 + 128: P1 = P1 + 128
2B 960 T = T + ( T > 2 ): FOR I = 0 TO 23: VTAB I + 1:
HTAB 1: E%(0,I) = P0: E%(1,I) = P1: E%(2,I) = P
0
52 970 FOR J = 1 TO 10: PRINT MID$( "...;";,T + 1,2
);: NEXT : PRINT MID$( "...;";,(T > 1) + 1,1);:
NEXT : RETURN
44 980 FOR P = 0 TO 1: HPLLOT XP + P, YP + P TO XP + D
X - P, YP + P TO XP + DX - P, YP + DY - P TO XP
+ P, YP + DY - P TO XP + P, YP + P: NEXT : RET
URN
7C 990 QF = 1: GOSUB 1000: RETURN
7F 1000 POKE 242,0: CALL 32777,AA
7D 1010 FOR I = 0 TO 23: FOR J = 0 TO 2: PRINT E%(J,
I): NEXT : NEXT : CALL 32780: RETURN
F9 1020 HOME : FOR I = 0 TO 23: FOR J = 0 TO 2: O = 1
27 - E%(J,I): IF O < 0 THEN O = O + 256
96 1030 E%(J,I) = O: FOR Q = 0 TO 6: T = INT ( O / 2 ):
PRINT CHR$( 46 + 13 * ( O - T * 2 ));: O = T:
NEXT : NEXT : IF I < 23 THEN PRINT
3B 1040 NEXT : RETURN

```

```

C7 1050 POKE 242,0: CALL 32777,0: GOSUB 1010: CALL 3
    2768,0,206,12: RETURN
DC 1060 CALL 32768,A,206,12
2B 1070 VTAB 1: HTAB 27: PRINT "ONE MOMENT";
DF 1080 CALL 32774,A: FOR I = 0 TO 23: FOR J = 0 TO
    2: INPUT " ";E%(J,I): NEXT : NEXT : CALL 3278
    0
7E 1090 HOME : FOR I = 0 TO 23: FOR J = 0 TO 2:O = E
    %(J,I)
92 1100 FOR Q = 0 TO 6:T = INT (O / 2): PRINT CHR$ (
    46 + 13 * (O - T * 2));O = T: NEXT : NEXT :
    IF I < 23 THEN PRINT
DC 1110 NEXT : HTAB 27: VTAB 1: PRINT SPC( 10): RETU
    RN
86 1120 GOSUB 560: GOSUB 700: VTAB 19: HTAB 10: PRIN
    T "INSERT BOX ";A;: GOSUB 1160: IF C = 206 T
    HEN 1150
9E 1130 GOSUB 700: IF A < 20 THEN CALL 32783,A
86 1140 POKE 242,0: CALL 32777,A: PRINT A$: CALL 327
    80: FOR A = A TO 20: GOSUB 500: NEXT
F0 1150 GOSUB 660: RETURN
ED 1160 PRINT " -REALLY/";
AB 1170 C = PEEK (49152): IF C < 128 THEN 1170
IF 1180 POKE 49168,0: IF C = 206 THEN PRINT "NO": RE
    TURN
82 1190 IF C = 217 THEN PRINT "YES": RETURN
6C 1200 GOTO 1170
70 1210 GOSUB 560: GOSUB 700: VTAB 19: HTAB 10: PRIN
    T "DELETE BOX ";A;: GOSUB 1160: IF C = 206 T
    HEN 1240
1E 1220 GOSUB 700: IF A < 20 THEN CALL 32786,A
AF 1230 POKE 242,0: CALL 32777,20: PRINT A$: CALL 32
    780: FOR A = A TO 20: GOSUB 500: NEXT
EE 1240 GOSUB 660: RETURN
90 1250 F$ = "SAVE": GOSUB 1270: IF LEN (N$) < > 0 T
    HEN PRINT : PRINT D$;"BSAVE ";N$;"",A$84C4,L$
    5E8"
F6 1260 GOTO 590
52 1270 HOME : TEXT : VTAB 2: PRINT "ESC TO CANCEL,
    RETURN FOR CATALOG"
A1 1280 PRINT : PRINT F$;" FILENAME: ";N$ = ""
4B 1290 GOSUB 1320: IF C$ = CHR$ (27) THEN N$ = "":
    RETURN
CE 1300 IF N$ = "" THEN PRINT : PRINT D$;"CATALOG":
    GOTO 1280
DB 1310 RETURN
AC 1320 T = FRE (0): GET C$: IF C$ = CHR$ (13) OR C$
    = CHR$ (27) THEN RETURN
BA 1330 IF C$ < > CHR$ (127) AND C$ < > CHR$ (8) THE
    N N$ = N$ + C$: PRINT C$;: GOTO 1320

```

```

CE 1340 IF N$ = "" THEN 1320
29 1350 HTAB LEN (N$) + 14: PRINT " "; HTAB LEN (N$
    ) + 14: IF LEN (N$) = 1 THEN N$ = "": GOTO 1
    320
8A 1360 N$ = LEFT$ (N$, LEN (N$) - 1): GOTO 1320
0C 1370 F$ = "LOAD": GOSUB 1270: IF LEN (N$) < > 0 T
    HEN PRINT : PRINT D$;"BLOAD ";N$;"A$84C4"
01 1380 GOTO 590
EA 1390 PRINT : PRINT "AN ERROR HAS OCCURRED"
5E 1400 PRINT "MAKE SURE THAT YOU HAVE A FORMATTED"
F9 1410 PRINT " DISK IN THE DRIVE"
49 1420 PRINT : PRINT "PRESS ANY KEY TO CONTINUE"
C2 1430 GET W$: GOSUB 590: GOTO 170
BE 1440 GOSUB 700: VTAB 19: HTAB 10: PRINT "QUIT ANI
    MATOR"; GOSUB 1160: IF C = 217 THEN MQ = 1:
    RETURN
F6 1450 GOSUB 660: RETURN
A7 1460 CALL 32736: GOTO 520
BD 1470 GOSUB 700: VTAB 19: HTAB 9: PRINT "CLEAR ALL
    BOXES"; GOSUB 1160: IF C = 206 THEN 1490
70 1480 GOSUB 700: GOSUB 650: GOSUB 520
07 1490 GOSUB 660: RETURN
36 1500 AB = F: RETURN
9A 1510 AE = F: RETURN
40 1520 A = AB: QF = 0: AR = 0: AX = 0
BD 1530 CALL 32771,A0,60: CALL 32768,A,AX,60:A0 = AX
    : IF QF = 1 THEN RETURN
0B 1540 C = PEEK (49152): IF C > 128 THEN POKE 49168
    ,0: GOSUB 1600
E3 1550 FOR I = 0 TO AP * 5: NEXT :AR = AR + AS: IF
    AR > 259 THEN AR = 0
5C 1560 IF AR < 0 THEN AR = 259
8A 1570 AX = 2 * INT (AR / 2): IF AE > AB THEN A = A
    + 1: IF A > AE THEN A = AB
3B 1580 IF AE < AB THEN A = A - 1: IF A < AE THEN A
    = AB
8E 1590 GOTO 1530
61 1600 IF C < > 160 THEN 1630
96 1610 IF PEEK (49152) < 128 THEN 1610
03 1620 POKE 49168,0: RETURN
36 1630 IF C = 136 THEN GOSUB 1670: GOTO 710
08 1640 IF C = 149 THEN GOSUB 1660: GOTO 710
43 1650 QF = 1: RETURN
1B 1660 AS = AS + (AS < 15): RETURN
D2 1670 AS = AS - (AS > - 15): RETURN
5A 1680 AP = AP - (AP > 0): RETURN
01 1690 AP = AP + (AP < 150): RETURN
BC 1700 DATA QUIT,DRAW,MOVE,ERASE,CLEAR,SAVE,INVERT,
    UPDATE,REVERT

```


- 90 1710 DATA BLACK1, GREEN, PURPLE, WHITE1, BLACK2, ORANGE, BLUE, WHITE2
- 80 1720 DATA EDIT FRAME, INSERT FRAME, DELETE FRAME, RANGE BOTTOM, RANGE TOP
- E6 1730 DATA ANIMATE, FASTER SPEED, SLOWER SPEED, LESS PAUSE
- A0 1740 DATA MORE PAUSE, LOAD, SAVE, CLEAR, INVERT, QUIT

Hi-Res Graphics Aid Routines

Jon Hylands

This handy utility makes it easy to perform sophisticated operations on Apple high-resolution graphics screens: inverting screens, copying screens, superimposing one screen on another, and more. It works on any Apple II series computer using either DOS 3.3 or ProDOS.

Like most personal computers, Apple II series machines can display high-resolution color graphics. There are many commercial programs, like *Fantavision* and *Dazzle Draw*, to name just two, that let you draw, save, and reload hi-res screens.

But few of these programs let you easily perform complex operations such as inverting an entire hi-res screen or superimposing one screen on another. "Hi-Res Graphics Aid" fills that gap. Though the program uses machine language for speed, you don't need to know ML to use it.

Type in and save the program to disk, then run it. The screen prompts are self-explanatory. Keep in mind that this is not a general-purpose drawing or design program. Instead, it performs large-scale tasks on existing graphics screens. Since the Apple can store two hi-res screens in memory at a time, most operations let you act on either screen 1 or screen 2.

When you run Graphics Aid, it displays a main menu of six selections. From this menu you can display a screen, edit a screen, load a screen, save a screen, display a disk catalog, or quit. The current selection is highlighted in reverse video. To choose a different selection, press the up-arrow or down-arrow key (on the Apple II+, Ctrl-K or Ctrl-J, respectively), then press Return. Here's a brief description of the options:

Display screen. Enter 1 to display screen 1; 2 for screen 2.

Edit screen. This option displays a second menu with the following options:

- *Display screen.* Enter 1 or 2.
- *Invert screen.* Enter 1 or 2.
- *Copy screen.* Enter 1 to copy screen 1 to screen 2, or vice versa.

- *Superimpose screen.* Enter 1 to superimpose screen 1 on screen 2, and vice versa. Then choose the mode by pressing a number key 1–3. Mode 1 is ORA mode—every pixel that's turned on in either screen remains on. Mode 2 is AND mode, in which only pixels that are on in both screens remain on. In Mode 3 (XOR), every pixel that's turned on in both screens will be turned off, and vice versa.
- *Color screen.* Choose screen 1 or 2, then enter a color number 0–7.
- *Flip high bits.* Choose screen 1 or 2, then choose the mode by pressing a number key 1–3. Mode 1 sets the high bits, mode 2 clears them, and mode 3 inverts them (on bits are turned off, and vice versa).
- *Swap screens.* Swap the contents of screen 1 and screen 2.
- *Return to command menu.*

Load to screen. Choose screen 1 or 2; then select drive 1 or 2, and enter the filename of the graphics file you wish to load.

Save screen. Choose screen 1 or 2; then select drive 1 or 2, and enter the filename you wish to use when saving the graphics screen to disk.

Catalog. Display a disk catalog.

Quit. Exit to BASIC.

Hi-Res Graphics Aid

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
B2 10 BA = 32768: FOR I = BA TO BA + 212: READ A:CK
    = CK + A: POKE I,A: NEXT : REM LOAD HR.CODE
CA 20 IF CK < > 31397 THEN PRINT "ERROR IN DATA STAT
    EMENTS.": STOP
F4 30 DATA 76,18,128,76,33,128,76,55
A0 40 DATA 128,76,80,128,76,115,128,76
E1 50 DATA 151,128,166,255,173,80,192,173
E7 60 DATA 82,192,173,87,192,189,83,192
7D 70 DATA 96,166,255,189,195,128,133,251
35 80 DATA 32,186,128,177,250,73,255,145
5F 90 DATA 250,32,177,128,208,245,96,166
68 100 DATA 255,189,195,128,133,251,189,198
F9 110 DATA 128,133,253,32,186,128,177,250
BE 120 DATA 145,252,32,177,128,208,247,96
FD 130 DATA 166,255,189,195,128,133,251,189
07 140 DATA 198,128,133,253,166,254,189,201
05 150 DATA 128,141,105,128,32,186,128,177
45 160 DATA 250,17,252,145,252,32,177,128
```



```

21 170 DATA 208,245,96,166,255,189,195,128
07 180 DATA 133,251,166,254,189,205,128,141
ED 190 DATA 141,128,189,209,128,141,142,128
CA 200 DATA 32,186,128,177,250,9,128,145
79 210 DATA 250,32,177,128,208,245,96,169
82 220 DATA 32,133,251,10,133,253,32,186
92 230 DATA 128,177,250,72,177,252,145,250
08 240 DATA 104,145,252,32,177,128,208,241
DE 250 DATA 96,200,208,5,230,251,230,253
73 260 DATA 202,96,160,0,132,250,132,252
EA 270 DATA 162,32,96,0,32,64,0,64
23 280 DATA 32,0,17,49,81,0,9,41
38 290 DATA 73,0,128,127,128
16 300 TEXT : HOME : PRINT : PRINT CHR$ (4); "PR#0":
      PRINT : REM INITIALIZATION
2A 310 D$ = CHR$ (4):BE$ = CHR$ (7):E$ = CHR$ (27):L
      $ = "-----":D = 1
F4 320 READ L: DIM X(L),TI$(L,13)
52 330 FOR J = 1 TO L: READ X(J): FOR I = 1 TO X(J):
      READ TI$(J,I): NEXT : NEXT
57 340 DATA 2,6,DISPLAY SCREEN,SCREEN EDITOR,LOAD SC
      REEN,SAVE SCREEN,CATALOG,QUIT
E4 350 DATA 8,DISPLAY SCREEN,INVERT SCREEN,COPY SCRE
      EN,SUPERIMPOSE SCREEN,COLOR SCREEN,FLIP HI BI
      TS,SWAP SCREENS,COMMAND MENU
1B 360 READ X: DIM ER$(X): FOR I = 1 TO X: READ ER$(
      I): NEXT
CC 370 DATA 13,,,WRITE PROTECTED,,FILE NOT FOUND,VO
      LUME MISMATCH,I/O ERROR,DISK FULL,FILE LOCKED
      ,SYNTAX ERROR,,FILE TYPE MISMATCH
09 380 REM COMMAND MENU
81 390 HOME : TEXT :J = 1:M$ = "HI.RES COMMAND MENU"
      : GOSUB 640
51 400 IF I = X(J) THEN VTAB 10 + X(J): END
50 410 ON I GOSUB 890,840,1290,1360,1430
9B 420 GOTO 390
4B 430 REM GET A KEYSTROKE
22 440 A = 0: GET A$: IF A$ = E$ THEN POP : RETURN
23 450 A = VAL (A$): RETURN
68 460 REM CENTER MESSAGE
C0 470 VTAB V: HTAB ( INT ((40 - LEN (M$)) / 2) + 1)
      : PRINT M$: RETURN
F6 480 REM DRAW A LINE
66 490 VTAB V: FOR I = 1 TO 4: PRINT L$;: NEXT : RET
      URN
79 500 REM GET DRIVE
84 510 PRINT "DRIVE : ";D; CHR$ (8);
10 520 GOSUB 440: IF A$ = CHR$ (13) THEN A = 1
82 530 IF A < 1 OR A > 2 THEN 520
7E 540 D = A: RETURN

```

```

7D 550 REM   GET PAGE
ED 560 GOSUB 440: IF A < 0 OR A > 2 THEN 560
E4 570 P = A: RETURN
0A 580 REM   ASK 'ARE YOU SURE ? '
D6 590 PRINT "ARE YOU SURE ? Y"; CHR$ (8);
67 600 GET A$: IF A$ = "N" OR A$ = E$ THEN PRINT A$;
      : POP : RETURN
24 610 IF A$ = CHR$ (13) OR A$ = "Y" THEN RETURN
16 620 GOTO 600
21 630 REM   CUSTOM MENU ROUTINE
F1 640 V = 2: GOSUB 490
F3 650 V = 4: GOSUB 470
76 660 V = 6: GOSUB 490
21 670 PRINT : VTAB 9
AB 680 FOR I = 1 TO X(J): HTAB 2: PRINT TI$(J,I): NE
      XT
3A 690 I = 1: VTAB 24: CALL - 868
E7 700 VTAB I + 8: HTAB 2: INVERSE : PRINT TI$(J,I):
      NORMAL
54 710 A = PEEK ( - 16384): IF A < 128 THEN 710
C4 720 POKE - 16368,0:A = A - 128
CF 730 IF A = 21 OR A = 10 THEN 770
CE 740 IF A = 8 OR A = 11 THEN 800
A9 750 IF A = 13 THEN RETURN
AB 760 GOTO 710
89 770 VTAB I + 8: HTAB 2: PRINT TI$(J,I)
C8 780 IF I + 1 > X(J) THEN I = 1: GOTO 700
BD 790 I = I + 1: GOTO 700
7C 800 VTAB I + 8: HTAB 2: PRINT TI$(J,I)
AE 810 IF I = 1 THEN I = X(J): GOTO 700
31 820 I = I - 1: GOTO 700
99 830 REM   SCREEN EDITOR
8E 840 HOME : TEXT :J = 2:M$ = "SCREEN EDITOR": GOSU
      B 640
EE 850 IF I = X(J) THEN RETURN
D3 860 ON I GOSUB 890,930,970,1020,1100,1190,1260
27 870 GOTO 840
65 880 REM   DISPLAY SCREEN
3A 890 VTAB 23: PRINT : PRINT "DISPLAY SCREEN : ";
70 900 GOSUB 560: IF A = 0 THEN RETURN
9D 910 POKE 255,P: CALL BA: GOTO 900
6D 920 REM   INVERT SCREEN
86 930 VTAB 23: PRINT : PRINT "INVERT SCREEN : ";
78 940 GOSUB 560: IF A = 0 THEN RETURN
28 950 POKE 255,P: CALL BA + 3: RETURN
20 960 REM   COPY SCREEN
14 970 VTAB 22: PRINT : PRINT "COPY SCREEN ";: GOSUB
      560: IF A = 0 THEN RETURN
73 980 POKE 255,P: PRINT P;" TO ";3 - P
68 990 GOSUB 590

```

```

AC 1000 CALL BA + 6: RETURN
C5 1010 REM SUPERIMPOSE SCREEN
14 1020 VTAB 21: PRINT : PRINT "SUPERIMPOSE SCREEN "
    ;: GOSUB 560: IF A = 0 THEN RETURN
72 1030 POKE 255,P: PRINT P;" TO ";3 - P
6C 1040 PRINT "1 : ORA 2 : AND 3 : EOR CHOOSE :
    ";
4A 1050 GOSUB 440
7F 1060 IF A < 1 OR A > 3 THEN 1050
31 1070 PRINT A: POKE 254,A: GOSUB 590
E4 1080 CALL BA + 9: RETURN
AA 1090 REM COLOR SCREEN
E2 1100 VTAB 21: PRINT : PRINT "COLOR SCREEN : ";: G
    OSUB 560: IF A = 0 THEN RETURN
F1 1110 PRINT P: PRINT "COLOR : ";
A4 1120 GET A$: IF A$ = E$ THEN RETURN
98 1130 IF A$ = "0" THEN C = 0: GOTO 1150
08 1140 C = VAL (A$): IF C < 1 OR C > 7 THEN 1120
72 1150 PRINT C: GOSUB 590
98 1160 POKE 230,32 * P: HCOLOR= C: HPLLOT 0,0: CALL
    62454
EF 1170 RETURN
C9 1180 REM FLIP HI BITS
87 1190 VTAB 21: PRINT : PRINT "FLIP HI BITS ON SCRE
    EN : ";: GOSUB 560: IF A = 0 THEN RETURN
46 1200 PRINT P: POKE 255,P: PRINT "1 : SET 2 : CLE
    AR 3 : FLIP CHOOSE : ";
3E 1210 GOSUB 440
43 1220 IF A < 1 OR A > 3 THEN 1210
34 1230 PRINT A;: POKE 254,A: GOSUB 590
5C 1240 CALL BA + 12: RETURN
55 1250 REM SWAP SCREENS
3A 1260 VTAB 23: PRINT : GOSUB 590
98 1270 CALL BA + 15: RETURN
FD 1280 REM LOAD SCREEN
AE 1290 VTAB 20: PRINT : PRINT "LOAD TO SCREEN : ";:
    GOSUB 560: IF A = 0 THEN RETURN
8C 1300 PRINT P: GOSUB 510: IF A = 0 THEN RETURN
15 1310 PRINT D: INPUT "FILENAME : ";F$
C3 1320 IF F$ = "" THEN RETURN
35 1330 VTAB 1: PRINT : PRINT D$;"BLOAD";F$;"D";D;"
    ,A";P * 8192
E7 1340 RETURN
97 1350 REM SAVE SCREEN
20 1360 VTAB 20: PRINT : PRINT "SAVE SCREEN : ";: GO
    SUB 560: IF A = 0 THEN RETURN
A8 1370 PRINT P: GOSUB 510: IF A = 0 THEN RETURN
31 1380 PRINT D: INPUT "FILENAME : ";F$
DF 1390 IF F$ = "" THEN RETURN

```



```
11 1400 VTAB 1: PRINT : PRINT D$;"BSAVE";F$;"D";D;"
,A";P * 8192;"",L8192"
DD 1410 RETURN
F6 1420 REM CATALOG DISK
36 1430 VTAB 23: PRINT : GOSUB 510: IF A = 0 THEN RE
TURN
22 1440 HOME :M$ = "CATALOG OF DRIVE " + STR$ (D):V
= 1: GOSUB 470
95 1450 V = 2: GOSUB 490
CA 1460 POKE 34,2: PRINT : PRINT D$"CATALOG,D"D
91 1470 V = 2:M$ = " PRESS A KEY..." : GOSUB 470
3B 1480 VTAB 2: HTAB 27: GET T$: POKE 34,0: RETURN
```

Apple Hi-Res Screen Dump

Mark Russinovich

You can easily dump high-resolution graphics pictures onto a dot-matrix printer with this efficient machine language utility. It's also an ideal way to add a screen dump option to your own BASIC programs. "DUMP," the accompanying program, requires an Apple IIe or II+ computer with at least 48K RAM and an Epson or Epson-compatible printer as well as an Epson or Epson-compatible parallel interface card in slot 1. For both DOS 3.3 and ProDOS.

Have you ever wanted to print out an image on the hi-res screen? This can be useful for inserting graphs or charts directly into text, or just for saving interesting pictures and mathematical plots. With "DUMP," the program below, you can do all these things with minimal effort.

Using DUMP

To get started, type in Program 1 using the "Apple MLX" machine language entry program found in Appendix C.

Note: Before loading Apple MLX, type this line:

HIMEM: 35844 when using DOS 3.3

or

HIMEM: 35840 when using ProDOS

Press Return. Now load MLX and respond to the starting and ending address prompts which appear:

Starting address: 9000

Ending address: 91DF

If you enter DUMP in more than a single session, make sure you type the HIMEM: 35844 or HIMEM: 35840 each time before loading MLX.

When you plan to use DUMP with ProDOS, you'll need to make one change to the listing you see at the end of the article. Type in the following line in place of line 9010 in the listing:

9010: 00 85 73 A9 8C 85 74 60 5F

After you finish typing in the data, use MLX to save it to disk with the name DUMP. To install DUMP in memory for later use, just type BRUN DUMP. It loads itself into memory, protects itself from Applesoft by resetting HIMEM, and changes the ampersand (&) vector to allow access from Applesoft.

Program 2 makes it easy to catalog disks, load hi-res pictures, view the pictures, and dump them on the printer. Just select the function you want from the menu. When you choose to print the hi-res screen, the program asks you to specify the size of the printout. There are nine sizes, ranging from a small block to a full page. (Owners of Epson MX-series printers should note that sizes 2, 3, 6, and 7 will *not* work with their printers. These sizes use codes available only on Epson's newer FX and RX models.) Next, you'll be asked for a tab value. This lets you position the picture exactly where you want it. Specify the tab value in pica characters (there are ten pica characters per inch), making sure the value doesn't exceed the width of the page minus the width of the picture. Otherwise, the picture might be cut off at the edge of the page or wrap around to the middle. If you enter a tab value of zero, DUMP automatically centers the picture on the page.

Table of DUMP Sizes

Size	Width Pica	Width Elite	Width Inch	Height Char	Height Inch
1	24	28	2.31	14	2.31
2	35	42	3.50	14	2.31
3	35	42	3.50	32	5.31
4	47	56	4.62	14	2.31
5	47	56	4.62	32	5.31
6	58	69	5.75	32	5.31
7	58	69	5.75	48	8.00
8	70	84	7.00	32	5.31
9	70	84	7.00	48	8.00

To embed a picture within the text of a document, you should leave room for the pictures in your document by changing the margins. For your convenience, the accompanying table shows the widths and heights of all nine print sizes.

After printing out your document, rewind the paper to position the printhead about one line above the space you left for the picture. Then run Program 2 and request the size and tab value you planned for. This procedure might take a little practice before you can place a picture exactly where you want it.

Note that DUMP sets the printer to all of its default values after running. If you were using a special mode or typeface, you'd have to restore that mode after running DUMP.

DUMP with Other Programs

DUMP is especially handy when used with graphing and drawing programs, and for this reason you may want to add it to programs of your own. To do this, add a line at the beginning of the program similar to this:

```
10 PRINT CHR$(4);"BRUN DUMP"
```

Later in the program, add a screen dump option to your menu. Prompt the user for size and tab values, then enter this command:

&P,S,T

where P specifies hi-res page (1 or 2), S is the size (1-9, but remember the MX-series limitation mentioned above), and T is the tab value. Program 2 is an example of how this is done. Numbers, variables, or expressions can be used in the command. For instance, to print out hi-res page 1, with a size of 3 and a tab of 15, this form could be used:

```
10 A=15
```

```
20 & 1,A/5,A
```

After DUMP has finished printing the picture, it returns control to Applesoft and the program continues running.

The ampersand command can also be entered in immediate mode.

Program 1. MLX Data for DUMP

For mistake-proof program entry, use "Apple MLX" (Appendix C) to type in this program. (Note the one-line change listed in the article for ProDOS users.)

```
START ADDRESS: 9000
```

```
END ADDRESS: 91DF
```

```
9000: A9 4C 8D F5 03 A9 18 8D 96
```

```
9008: F6 03 A9 90 8D F7 03 A9 9F
```

```
9010: 00 85 73 A9 8C 85 74 60 5F
```

```
9018: 20 5B 91 20 F8 E6 E0 01 7A
```

```

9020: F0 07 E0 02 F0 07 20 C9 65
9028: DE A9 20 D0 02 A9 40 85 F0
9030: E6 20 BE DE 20 F8 E6 E0 26
9038: 0A B0 1A CA 8E D5 91 20 89
9040: BE DE 20 F8 E6 E0 00 D0 97
9048: 06 AC D5 91 BE 6B 91 8E C0
9050: DB 91 4C 58 90 20 C9 DE 4A
9058: A0 00 84 F9 84 FA 84 EF 03
9060: 84 FB 84 FD 84 FE 84 FF 5C
9068: AC D5 91 A9 1B 20 62 91 D1
9070: A9 33 20 62 91 B9 74 91 4B
9078: 20 62 91 B9 7D 91 8D D6 34
9080: 91 B9 86 91 8D D7 91 B9 6B
9088: 8F 91 8D D8 91 AA CA 8E 70
9090: D9 91 B9 98 91 8D DA 91 CD
9098: 98 0A AB B9 AB 91 85 06 D5
90A0: B9 A9 91 85 07 A0 00 B1 FF
90A8: 06 99 DC 91 C8 C0 03 D0 08
90B0: F6 A9 0A 20 62 91 AC DB 89
90B8: 91 F0 08 A9 20 20 62 91 52
90C0: 88 D0 FB A0 00 A9 1B 20 80
90C8: 62 91 B9 DC 91 20 62 91 E7
90D0: C8 C0 03 D0 F5 A5 FB 85 B7
90D8: FC A5 FC C9 C0 B0 28 A6 DD
90E0: F9 A4 FA 20 11 F4 A4 EF 1F
90E8: B1 26 A4 FF 39 A1 91 F0 65
90F0: 16 AD DA 91 A4 FD F0 0B 07
90F8: AE D6 91 4A CA D0 FC 88 1A
9100: 4C F6 90 05 FE 85 FE E6 5C
9108: FC E6 FD A5 FD CD D8 91 E7
9110: D0 C7 A5 FE AC D7 91 20 3A
9118: 62 91 88 D0 FA A9 00 85 F2
9120: FE 85 FD E6 FF A5 FF C9 B2
9128: 07 D0 06 E6 EF A9 00 85 DD
9130: FF E6 F9 D0 02 E6 FA A5 A0
9138: F9 C9 18 D0 98 A5 FA F0 1D
9140: 94 A9 00 85 EF 85 F9 85 7F
9148: FA A5 FB 6D D9 91 85 FB C4
9150: C9 C0 B0 03 4C B1 90 20 39
9158: 5B 91 60 A9 1B 20 62 91 E3
9160: A9 40 AE C1 C1 30 FB 8D AE
9168: 90 C0 60 1D 16 16 11 11 1E
9170: 0A 0A 05 05 15 15 12 15 42
9178: 12 12 12 12 12 01 01 02 25
9180: 01 02 02 03 02 03 01 03 36
9188: 03 01 01 05 05 03 03 07 1F
9190: 07 03 07 03 03 02 03 02 31
9198: 40 40 60 40 60 60 70 60 C1
91A0: 70 01 02 04 08 10 20 40 BD
91A8: BA 91 BD 91 C0 91 C3 91 C3
91B0: C6 91 C9 91 CC 91 CF 91 CB

```

```

91B8: D2 91 4C 18 01 5A 48 03 B9
91C0: 5A 48 03 4B 18 01 4B 18 AB
91C8: 01 5A 78 05 5A 78 05 4C 6D
91D0: 48 03 4C 48 03 A2 D6 C9 01
91D8: A5 8A A0 C5 C8 FF A0 20 89

```

Program 2. DUMP Example

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

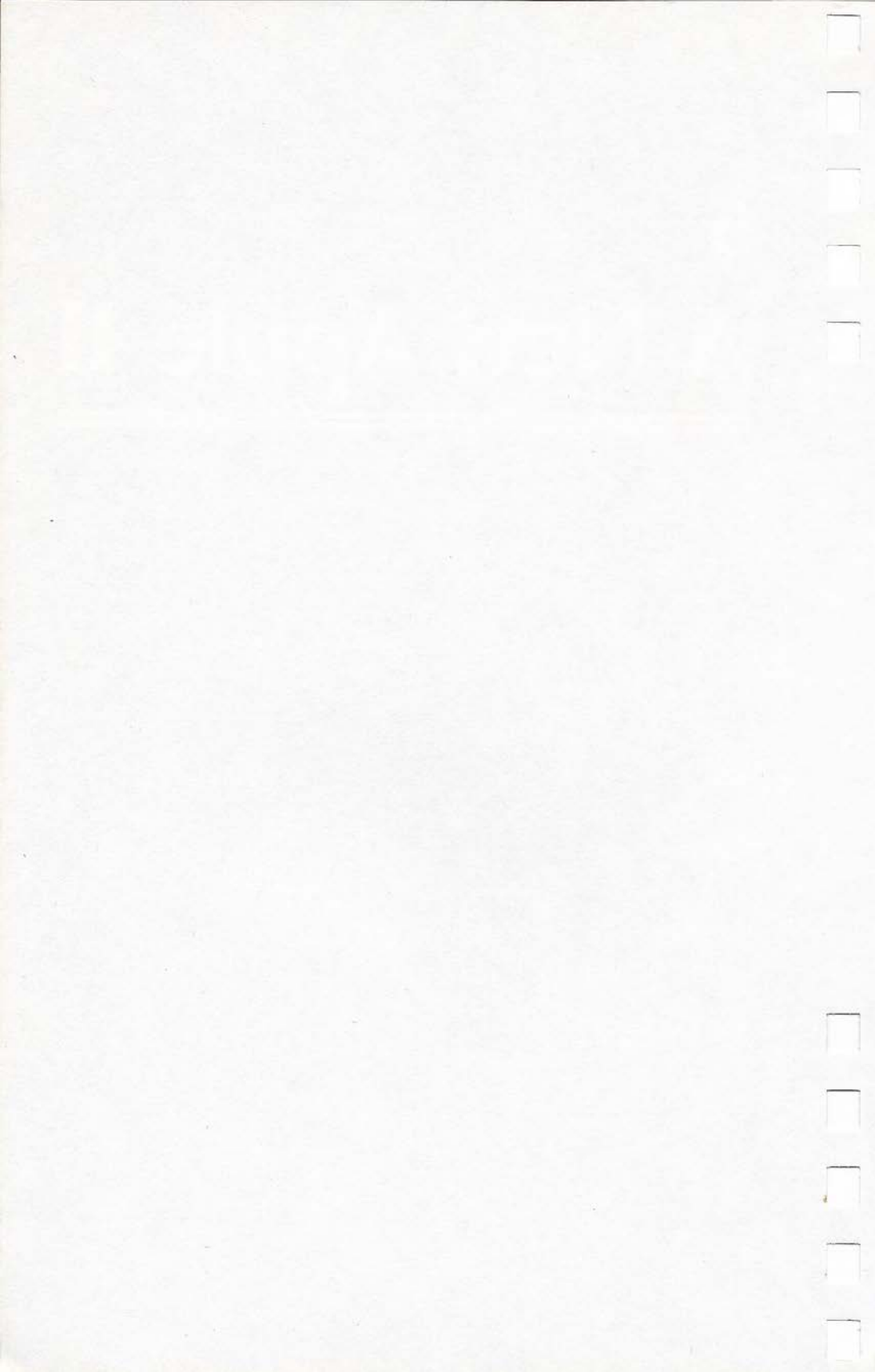
```

81 10 ONERR GOTO 40
52 20 D$ = CHR$ (4)
05 30 PRINT D$"BRUN DUMP"
55 40 TEXT : HOME
13 50 HTAB 9: PRINT "*****": HTAB 9:
    PRINT "*" *: HTAB 9: PRINT
    "* APPLE HI-RES DUMP *": HTAB 9: PRINT "*"
    *: HTAB 9: PRINT "*****"
*****
1F 60 PRINT : PRINT : PRINT : HTAB 12: PRINT "ENTER
CHOICE:": HTAB 12: PRINT "-----": PRIN
T : HTAB 12: PRINT "1) CATALOG": PRINT : HTAB
12: PRINT "2) LOAD SCREEN"
8A 70 PRINT : HTAB 12: PRINT "3) VIEW SCREEN": PRINT
: HTAB 12: PRINT "4) PRINT SCREEN": PRINT : H
TAB 12: PRINT "5) QUIT"
D7 80 VTAB 22: HTAB 12: GET A$: IF A$ = "1" THEN 140
2A 90 IF A$ = "2" THEN 160
F6 100 IF A$ = "3" THEN 170
74 110 IF A$ = "4" THEN 200
48 120 IF A$ = "5" THEN END
D0 130 PRINT CHR$ (7): GOTO 80
60 140 PRINT : HOME : PRINT D$"CATALOG"
E3 150 PRINT : PRINT "PRESS ANY KEY:": GET A$: GOTO
40
DD 160 PRINT : VTAB 22: INPUT "ENTER FILE NAME: ";FL
$: PRINT D$"BLOAD"FL$,A$2000": HOME : GOTO 4
0
81 170 POKE - 16302,0: POKE - 16297,0: POKE - 16304,
0: POKE - 16368,0
87 180 X = PEEK ( - 16384): IF X < 128 THEN 180
2B 190 POKE - 16368,0: TEXT : HOME : GOTO 40
11 200 PRINT : VTAB 22: INPUT "ENTER SIZE OF DUMP (1
-9): ";S: IF S < 1 OR S > 9 THEN 200
DD 210 VTAB 24: PRINT "(0=AUTO CENTER)": VTAB 23: H
TAB 1: INPUT "ENTER TAB SETTING: ";T: IF T <
0 OR T > 50 THEN 210
7A 220 POKE - 16302,0: POKE - 16297,0: POKE - 16304,
0: & 1,S,T: TEXT : HOME : GOTO 40

```


5

A New Apple II



Windows

Lee Swoboda

An Apple II computer is not a Macintosh, yet we're seeing more and more Mac-like software for the Apple II, II+, IIe, and IIc. Duplicating Macintosh-style windows—just one of the useful features of that machine's operating system—is simple with this program. For all Apple II series computers using either DOS 3.3 or ProDOS.

One of the features that makes the Macintosh so easy to use is its ability to open and close multiple *windows* on the screen. These windows—basically smaller screens superimposed on the main screen—can provide additional information, offer menu selections, or provide a notepad-style environment where you can enter and save text. Once the information has appeared or the menu item has been chosen, the window can be erased, letting you get on with the task at hand.

The Apple II series computers can create windows, too, even automatically save and restore text screens. With "Windows" at your disposal, you can open a window in an existing text screen and make it disappear, all without having to re-print the underlying screen. Windows easily simulates a Macintosh appearance in your own BASIC programs, letting you operate with as many as nine windows (ten if you count the main screen).

Machine Language the Easy Way

Though Windows is a machine language program, you don't need to know anything about machine language programming to enter or use it. Enter the program using the "Apple MLX" machine language entry program (see Appendix C). Be sure you read and understand the instructions for using MLX before you begin entering data. Before you load and run MLX to enter Windows (and each time you reload MLX if you enter the data in several sittings), you should enter the following line in direct mode (without a line number) and press Return:

HIMEM: 36352

Then load and run MLX. You'll be asked for a starting address and an ending address for the data you'll be entering. For Windows, the proper values are:

Starting address: 9200

Ending address: 954F

After you finish entering all the data, be sure to save at least one copy. To use the demonstration program (Program 2), you must use the filename WINDOWS when you save the Windows data. To load Windows, enter

BLOAD WINDOWS

Windows is now in memory, waiting. Simple.

But Windows does nothing all by itself. It must be used in conjunction with a BASIC program. Let's look at a demonstration of what Windows can do.

Showing Off

Type in and save Program 2, "Windows in BASIC." (Remember, you must use some name other than WINDOWS for this program.) This is a complete illustration of Windows' power, and works in either DOS 3.3 or ProDOS. If you're using the latter, however, you must make one change. Modify line 110 so that it reads

110 HIMEM: 33792

Make sure a copy of the WINDOWS file created from the data in Program 1 is on the same disk as Program 2, then type RUN. You'll see the main screen (Figure 1).

Figure 1. The Main Screen



Press the D key, then hit Return to run the demonstration. The computer will display the first window (Figure 2).

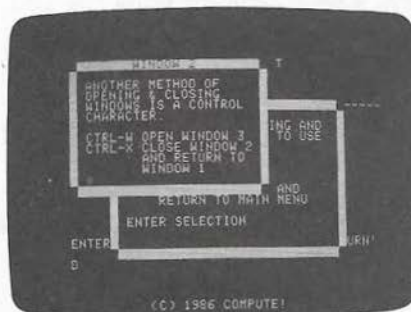
Figure 2. First Window Added



Window 1 is superimposed over the main screen, so parts of the latter still show around the solid white border of the window. The computer has saved the main screen to be re-stored later.

Press W to open window 2. This second window is also superimposed on the previous screen, so parts of both the main screen and window 1 show around its edges (Figure 3).

Figure 3. Another Window



Type Ctrl-W (Hold the Control key and press W). The computer places the third window over the ones already on the screen (Figure 4).

Figure 4. Window 3



Type your name (or anything else) in the blank on window 3 and press Return. The computer remembers what you typed, but closes window 3 and returns to window 2. Press Ctrl-X to close window 2, then hit the X key to close window 1. You're back where you started, with the main screen displayed.

In this demonstration, windows 1 and 2 were menus, but if a program allows you to type something on a window, it will be restored when you close the window.

Each time you opened a window, the computer saved the current screen to memory. Each time you closed a window, the computer restored the screen it had saved.

Now press Q and hit Return. The computer exits the program, printing the text you typed in window 3 on the screen as it says goodbye.

Inside Windows

The Apple's 40-column text screen is located at memory addresses 1024–2047. When you open a window, the machine language program Windows copies the data on the current text screen to a safe place above HIMEM and transfers it back when you close a window. Windows also stores information about the screen size and cursor location so that the computer remembers the exact screen arrangement when you close the window.

In Apple II-series computers, memory addresses 32–37 maintain information about the text screen:

Address	Contents
32	Left margin (default=0)
33	Width (default=40)
34	Top margin (default=0)
35	Bottom margin (default=24)
36	Horizontal cursor location
37	Vertical cursor location

Program 2 POKes values to these addresses to change the text screen characteristics. Take a close look at the listing. Though it's only a simple example, it shows how you can use Windows in your own programs.

Windows of Your Own

Lines 100–130 in Program 2 are mandatory to initialize the program parameters. *You must include these same lines (slightly modified) in your own program in order to use Windows.*

Line 110. The value of HIMEM in line 110 depends on the maximum number of windows you intend to use, and whether you're using DOS 3.3 or ProDOS. See Table 1 for the appropriate values.

Table 1. HIMEM Values

Maximum No. of Windows	DOS 3.3	ProDOS
1	36352	35840
2	35328	34816
3	34304	33792
4	33280	32768
5	32256	31744
6	31232	30720
7	30208	29696
8	29184	28672
9	28160	27648
10	27136	26624

Line 120. These POKes should be specified early in the program. (Table 2 shows the values which must be POKed into memory to open and close windows—you'll find the locations in line 120 listed in this table.) Of these three POKes, the only one which you'll need to change in your own pro-

gram is POKE 769,WMAX. Simply set WMAX to the maximum number of windows your program will allow.

Line 130. These POKES establish the default characteristics of the Apple II text screen. Take a look at the listing above (locations 32–37), and you'll see that the four POKES in this line set up these default values:

Left margin	0
Width	40
Top margin	0
Bottom margin	24

Enter these POKES in your own program just as you see them in line 130 of Program 2.

Opening Windows

Lines 300 and 310 in Program 2 are an example of the information you *must* provide to open a window. The POKES in line 300 define the size and location of the window, while the POKE and CALL in line 310 activates Windows. Each window is defined by POKEing the window characteristics before CALLing Windows with CALL 37376. For example, line 300 defines window 1 as having a left margin in column 5 (POKE 32,5), a width of 30 characters (POKE 33,30), a top margin at text line 4 (POKE 34,4), and a bottom margin at text line 19 (POKE 35,19).

Closing Windows

Line 430 is an example of closing a window. You need only to POKE 768,0 and CALL 37376—you don't need to redefine the window parameters. When Windows opens a window, it stores the window parameters, then automatically restores them when it closes the window.

Windows stores the parameters for each window in the normally unused space beginning at memory location 768 (\$0300 in hexadecimal). Table 2 lists the values stored at each address.

Each text screen is saved in a separate area above HIMEM, beginning with window 0 (the main screen), stored from memory addresses 36352–37376, and working downward.

When you close a window, the computer restores the original screen by POKEing the screen characteristics in locations 32–37 and moving the text screen from storage back to the text screen buffer at memory addresses 1024–2047. Note,

too, that with each window's margin and width values are stored the *previous* window's cursor positions. Thus, when you close a window, the cursor appears at the position it occupied *before* that window was opened.

Using Windows on your Apple II won't turn it into a Macintosh, but it can add some of the sophistication of the Macintosh to your BASIC programs. Open a window and see for yourself.

Table 2. Windows Variable Storage

Memory Address	Description	Monitor Address	Range
768	Direction of window movement	n/a	0=Open, 1=Close
769	Maximum number of windows	n/a	1-n
770	Current window number	n/a	0-10
771	Window 1, left margin	32	0-39
772	Window 1, width	33	1-40
773	Window 1, top margin	34	0-22
774	Window 1, bottom margin	35	1-24
775	Window 0, horizontal cursor position	36	0-39
776	Window 0, vertical cursor position	37	0-23
777	Window 2, left margin	32	0-39
778	Window 2, width	33	1-40
779	Window 2, top margin	34	0-22
780	Window 2, bottom margin	35	1-24
781	Window 1, horizontal cursor position	36	0-39
782	Window 1, vertical cursor position	37	0-23
783	Window 3, left margin	32	0-39
784	And so on		

Program 1. Windows

For mistake-proof program entry, use the "Apple MLX" (Appendix C) to type in this program.

START ADDRESS: 9200
END ADDRESS: 954F

```

9200: AD 59 AA 48 A5 D9 48 A5 F6
9208: 76 48 A9 02 85 76 A9 FF 29
9210: 85 D9 A9 BF 85 33 A9 00 EB
9218: 85 F3 4C 23 92 00 00 00 4D
9220: 92 00 08 A9 1D 85 85 A9 DD
9228: 92 A0 00 A2 05 20 2F 95 85

```



```

9230: AD 00 03 8D 1D 92 A9 00 EB
9238: 8D 1E 92 AD 1D 92 C9 01 A0
9240: D0 0A AD 1E 92 C9 00 D0 74
9248: 03 4C 43 93 AD 02 03 8D AC
9250: 1D 92 A9 00 8D 1E 92 AD 95
9258: 01 03 8D 1F 92 A9 00 8D 2B
9260: 20 92 EE 1D 92 D0 03 EE B6
9268: 1E 92 AD 1E 92 CD 20 92 77
9270: 30 0F D0 0A AD 1D 92 CD 01
9278: 1F 92 90 05 F0 03 4C ED 4E
9280: 94 20 7D 94 A9 20 8D 21 FA
9288: 92 A9 00 8D 22 92 AD 22 13
9290: 92 C9 00 30 0E D0 09 AD E7
9298: 21 92 C9 25 90 05 F0 03 FB
92A0: 4C DF 92 AD 21 92 8D B0 30
92A8: 92 AD 22 92 8D B1 92 AD F5
92B0: 25 00 8D 1D 92 A9 00 8D B4
92B8: 1E 92 AD 1F 92 8D CA 92 2C
92C0: AD 20 92 8D CB 92 AD 1D 11
92C8: 92 8D 0E 03 EE 1F 92 D0 76
92D0: 03 EE 20 92 EE 21 92 D0 52
92D8: 03 EE 22 92 4C 8E 92 20 8A
92E0: BB 94 A9 00 8D 21 92 A9 FD
92E8: 04 8D 22 92 AD 22 92 C9 C5
92F0: 07 30 0E D0 09 AD 21 92 48
92F8: C9 FF 90 05 F0 03 4C 3D CE
9300: 93 AD 21 92 8D 0E 93 AD 23
9308: 22 92 8D 0F 93 AD FF 07 E1
9310: 8D 1D 92 A9 00 8D 1E 92 37
9318: AD 1F 92 8D 28 93 AD 20 14
9320: 92 8D 29 93 AD 1D 92 8D E6
9328: FF 8D EE 1F 92 D0 03 EE 4F
9330: 20 92 EE 21 92 D0 03 EE C8
9338: 22 92 4C EC 92 20 58 FC 30
9340: 4C ED 94 AD 02 03 8D 1D CA
9348: 92 A9 00 8D 1E 92 AD 1D AF
9350: 92 D0 03 CE 1E 92 CE 1D 38
9358: 92 AD 1E 92 C9 00 30 09 D8
9360: D0 0A AD 1D 92 C9 00 B0 66
9368: 03 4C ED 94 20 7D 94 A9 F4
9370: 20 8D 21 92 A9 00 8D 22 E2
9378: 92 AD 22 92 C9 00 30 0E 7E
9380: D0 09 AD 21 92 C9 25 90 B0
9388: 05 F0 03 4C CA 93 AD 1F B2
9390: 92 8D 9B 93 AD 20 92 8D B1
9398: 9C 93 AD 02 03 8D 1D 92 E3
93A0: A9 00 8D 1E 92 AD 21 92 50
93A8: 8D B5 93 AD 22 92 8D B6 7E
93B0: 93 AD 1D 92 8D 25 00 EE C9
93B8: 1F 92 D0 03 EE 20 92 EE 6A

```

```

93C0: 21 92 D0 03 EE 22 92 4C D8
93C8: 79 93 20 BB 94 38 AD 1F 51
93D0: 92 E9 00 8D 1F 92 AD 20 53
93D8: 92 E9 04 8D 20 92 A9 00 BB
93E0: 8D 21 92 A9 04 8D 22 92 31
93E8: AD 22 92 C9 07 30 0E D0 44
93F0: 09 AD 21 92 C9 FF 90 05 C9
93F8: F0 03 4C 39 94 AD 1F 92 A2
9400: 8D 0A 94 AD 20 92 8D 0B 51
9408: 94 AD FF 91 8D 1D 92 A9 AF
9410: 00 8D 1E 92 AD 21 92 8D 2E
9418: 24 94 AD 22 92 8D 25 94 F9
9420: AD 1D 92 8D FF 07 EE 1F AB
9428: 92 D0 03 EE 20 92 EE 21 68
9430: 92 D0 03 EE 22 92 4C E8 03
9438: 93 AD 02 03 8D 1D 92 A9 B6
9440: 00 8D 1E 92 AD 1D 92 C9 8A
9448: 00 D0 1A AD 1E 92 C9 00 92
9450: D0 13 A9 00 85 20 A9 28 04
9458: 85 21 A9 00 85 22 A9 18 E1
9460: 85 23 4C ED 94 A9 00 85 4E
9468: 8A A9 06 AE 1E 92 AC 1D 9E
9470: 92 20 01 95 8E 1E 92 8C 03
9478: 1D 92 4C ED 94 AD 1D 92 65
9480: 8D 02 03 38 AD 1D 92 E9 C5
9488: 01 8D 1F 92 AD 1E 92 E9 97
9490: 00 8D 20 92 A9 00 85 8A 2D
9498: A9 06 AE 20 92 AC 1F 92 08
94A0: 20 01 95 8E 20 92 8C 1F 39
94A8: 92 18 A9 03 6D 1F 92 8D 21
94B0: 1F 92 A9 03 6D 20 92 8D 12
94B8: 20 92 60 AD 02 03 8D 1F D3
94C0: 92 A9 00 8D 20 92 A9 04 19
94C8: 85 8A A9 00 AE 20 92 AC 54
94D0: 1F 92 20 01 95 8E 20 92 FB
94D8: 8C 1F 92 38 A9 00 ED 1F 2E
94E0: 92 8D 1F 92 A9 92 ED 20 57
94E8: 92 8D 20 92 60 68 85 76 12
94F0: 68 85 D9 68 8D 59 AA A9 42
94F8: 8D 8D 01 02 A9 01 85 34 1D
9500: 60 85 89 84 87 86 88 A9 47
9508: 00 85 85 85 86 46 88 66 62
9510: 87 90 0D 18 A5 89 65 85 E9
9518: 85 85 A5 8A 65 86 85 86 9B
9520: 06 89 26 8A A5 88 05 87 FE
9528: D0 E3 A4 85 A6 86 60 85 37
9530: 86 84 87 A0 00 A9 00 91 F2
9538: 85 C8 D0 02 E6 86 8A D0 C9
9540: 04 C6 87 30 04 CA 4C 35 2C
9548: 95 60 00 CD 6A 8B F0 15 AB

```

Program 2. Windows in BASIC

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```

72 100 WMAX = 3
DF 110 HIMEM: 34304: REM SEE TABLE 1
38 120 POKE 768,0: POKE 769,WMAX: POKE 770,0
6F 130 POKE 32,0: POKE 33,40: POKE 34,0: POKE 35,24
5A 140 D$ = CHR$(4)
3E 150 PRINT D$"BLOAD WINDOWS"
50 160 HOME
F9 170 PRINT TAB(11)"APPLESOFT"
F0 180 PRINT
F3 190 PRINT TAB(13)"WINDOWS"
CD 200 PRINT : PRINT "-----"
      "-----": REM 40 DASHES
69 210 VTAB 24: PRINT TAB(11)"(C) 1986 COMPUTE!";
15 220 VTAB 9: HTAB 5: PRINT "(D) DEMONSTRATION OF W
      INDOWS"
6E 230 PRINT : PRINT TAB(5)"(Q) QUIT THE PROGRAM"
D4 240 VTAB 18: PRINT "ENTER YOUR SELECTION AND PRES
      S 'RETURN'"
06 250 VTAB 20: HTAB 1: CALL - 868: INPUT " ";A$
20 260 IF A$ = "D" THEN 300
46 270 IF A$ = "Q" THEN 780
21 280 GOTO 250
E1 290 REM WINDOW 1
3A 300 POKE 32,5: POKE 33,30: POKE 34,4: POKE 35,19
82 310 POKE 768,0: CALL 37376
51 320 GOSUB 860
AE 330 INVERSE : VTAB 5: HTAB 11: PRINT "WINDOW 1":
      NORMAL
1E 340 VTAB 7: HTAB 3: PRINT "ONE METHOD OF OPENING
      AND"
5A 350 HTAB 3: PRINT "CLOSING WINDOWS IS TO USE"
C3 360 HTAB 3: PRINT "A MENU."
28 370 VTAB 12: HTAB 3: PRINT "(W) OPEN WINDOW 2"
BD 380 HTAB 3: PRINT "(X) CLOSE WINDOW 1 AND"
8D 390 HTAB 7: PRINT "RETURN TO MAIN MENU"
BD 400 VTAB 16: HTAB 3: PRINT "ENTER SELECTION"
0D 410 VTAB 16: HTAB 19: GET A$
C6 420 IF A$ = "W" THEN 460
88 430 IF A$ = "X" THEN POKE 768,1: CALL 37376: GOTO
      250
18 440 GOTO 410
EB 450 REM WINDOW 2
F7 460 POKE 32,0: POKE 33,25: POKE 34,0: POKE 35,13
8F 470 POKE 768,0: CALL 37376
5E 480 GOSUB 860

```



```

26 490 VTAB 1: HTAB 9: INVERSE : PRINT "WINDOW 2": N
    ORMAL
8A 500 VTAB 3: HTAB 3: PRINT "ANOTHER METHOD OF"
30 510 HTAB 3: PRINT "OPENING & CLOSING"
37 520 HTAB 3: PRINT "WINDOWS IS A CONTROL"
89 530 HTAB 3: PRINT "CHARACTER."
1D 540 VTAB 8: HTAB 3: PRINT "CTRL-W OPEN WINDOW 3"
E8 550 HTAB 3: PRINT "CTRL-X CLOSE WINDOW 2"
E3 560 HTAB 10: PRINT "AND RETURN TO"
BC 570 HTAB 10: PRINT "WINDOW 1"
5F 580 VTAB 12: HTAB 3: GET A$
51 590 IF A$ = CHR$ (23) THEN 630
18 600 IF A$ = CHR$ (24) THEN POKE 768,1: CALL 37376
    : GOTO 410
9B 610 GOTO 580
F7 620 REM WINDOW 3
62 630 POKE 32,15: POKE 33,25: POKE 34,9: POKE 35,21
8B 640 POKE 768,0: CALL 37376
5A 650 GOSUB 860
58 660 VTAB 10: HTAB 9: INVERSE : PRINT "WINDOW 3":
    NORMAL
D2 670 VTAB 12: HTAB 3: PRINT "THIS IS THE LAST WIN-
    "
E6 680 HTAB 3: PRINT "DOW. TYPE SOMETHING"
EC 690 HTAB 3: PRINT "BELOW, THEN PRESS"
8F 700 HTAB 3: PRINT "'RETURN' TO CLOSE"
EF 710 HTAB 3: PRINT "WINDOW 3 AND RETURN"
6D 720 HTAB 3: PRINT "TO WINDOW 2."
F3 730 VTAB 19: HTAB 3: PRINT ".....
    "
EE 740 VTAB 19: HTAB 3: INPUT "":B$
96 750 POKE 768,1: CALL 37376
A6 760 GOTO 580
69 770 REM QUIT
5A 780 HOME
CD 790 IF B$ = "" THEN B$ = "NOTHING"
DA 800 VTAB 10: PRINT "YOU ENTERED"
4E 810 VTAB 12: PRINT CHR$ (34)B$ CHR$ (34)
21 820 VTAB 14: PRINT "ON WINDOW 3"
A7 830 VTAB 20: PRINT "GOODBYE" CHR$ (7) CHR$ (7)
98 840 END
36 850 REM BORDER
0B 860 BL$ = "
    ": REM 40 SPACES
7D 870 I = PEEK (770)
13 880 IF I = 0 THEN RETURN
73 890 I = 771 + 6 * (I - 1)
7F 900 WL = PEEK (I):WW = PEEK (I + 1):WT = PEEK (I
    + 2):WB = PEEK (I + 3)

```

```
30 910 INVERSE
BE 920 HTAB 2: PRINT LEFT$ (BL$,WW - 2);
63 930 VTAB WB: HTAB 2: PRINT LEFT$ (BL$,WW - 2);
3A 940 FOR I = WT + 2 TO WB - 1
A7 950 VTAB I: HTAB 1: PRINT " "
20 960 VTAB I: HTAB WW: PRINT " ";
F3 970 NEXT I
DC 980 NORMAL
2B 990 RETURN
```

Apple II Pull-Down Menus

Lee Swoboda

Attractive pull-down menus and instruction screens are an important part of the easy-to-use Macintosh-like interface. This article and accompanying program show you how to add these features to any BASIC program. For all Apple II series computers using either DOS 3.3 or ProDOS.

Apple's Macintosh has forced programmers to reevaluate software for the venerable Apple II. Recent commercial Apple II programs go to some lengths to emulate the Macintosh's pull-down menus and icons to make the software less intimidating. Though no amount of programming magic can turn an Apple II into a Macintosh, by adding such things as pull-down menus, you can surely try.

Two programs are needed to let you add pull-down menus and instruction screens to your Applesoft BASIC programs.

1. A BASIC subroutine you can easily add to the end of any BASIC program.
2. A machine language (ML) routine that temporarily saves and later restores the text behind the pull-down menu.

Although BASIC takes several seconds to move an entire text screen, machine language performs the same task in an instant. Don't worry if you're unfamiliar with machine language. We've listed a BASIC filemaker program that automatically creates the ML routine for you.

Starting Out

To get "Pull-Down Menus" running, you need to type in and save both programs listed below. Program 1 is the ML routine to save and restore the text behind the menu. It must be entered using the "Apple MLX" machine language entry program found in Appendix C. Be sure you read and understand the instructions for using MLX before you begin entering the data. Before loading MLX to enter the data for the routine

(and before reloading MLX if you type the data in several sittings), you must enter the following line in direct mode (without a line number) and press Return:

HIMEM: 35891

Then load and run MLX. You'll be asked for a starting address and an ending address. Use the following values:

Starting address: 9033

Ending address: 91FF

When you finish entering the data, be sure to save a copy before you leave MLX. For the routine to work properly with the BASIC menu routine in Program 2, you must save the data with the filename MOVE.

Program 2 is an example BASIC program that demonstrates pull-down menus. It's designed to run with either DOS 3.3 or ProDOS. If you're using DOS 3.3, type the program exactly as shown. For ProDOS, though, change line 150 to:

150 HIMEM: 35840

Since this program loads the MOVE file from disk, be sure the disk in the drive contains a copy of that file before you run Program 2. Once you have it running, the program simulates a crude word processor with a screenful of text. You can type on the screen and move the cursor with the arrow keys (use Ctrl-J and Ctrl-K for the up and down cursor keys if you're using an Apple II+). When you press the Esc key, the pull-down menu appears. Move the selection cursor inside the menu with the cursor keys and choose a selection by pressing Return. Note that the text behind the menu is always restored correctly when you leave the menu.

Create Your Own Menus

The important part of the demonstration program is the sub-routine beginning at line 63000. This routine allows you to add pull-down menus to your own programs with a minimum of work: It generates the window shape and calls MOVE at the appropriate time. All you need to do is add lines 63000-63500 to the end of any BASIC program and follow the steps listed below.

1. Your program must BLOAD MOVE as shown in lines 180-190 before calling the ML routine.

2. Set HIMEM *immediately* (line 150) before you declare any strings or open any files. Use a value of 36914 for DOS 3.3 or 35840 for ProDOS.
3. Set the variable NN to equal the maximum number of items you will have in the largest menu (line 160). The menu subroutine automatically determines how many items are in each menu and adjusts the size of the menu window accordingly.
4. DIMension the string array MM\$ for the number of menu selection labels you need (line 170). Then fill each array element with a label string, either by READING string DATA as in lines 200–220 or by defining each string expressly—with statements like MM\$(1) = "Leave menu".
5. Define the string variable TITLE\$ as your menu title (line 470). The menu subroutine automatically centers the title for you.
6. Provide some means of branching to the rest of your program based on the value of the variable SELECT (line 480). This may be done with ON SELECT GOTO, as in this program, or with ON SELECT GOSUB or a series of IF-THEN statements.

Lines 690–850 of the program show how to use MOVE to add instructions to your programs without losing the original screen. In this case, Ctrl-I is used to request instructions.

Using a Mouse

If you have an Apple mouse, you can use it to call the menu and make selections. This requires several changes in the demonstration program.

First, delete lines 320, 330, and 63360–63460. Next, change lines 310, 450, and 63350 to the following:

```

E9 310 PRINT "PRESS ESC KEY OR MOUSE BUTTON FOR MENU
    ";
9B 450 GOTO 311
66 63350 HTAB 3: VTAB SELECT + 2: INVERSE : PRINT ">
    " CHR$(8);: NORMAL

```

Now add these lines:

```

D2 235 PRINT : HOME : PRINT D$"PR#2": PRINT CHR$(1)
    : PRINT D$"PR#0"
0A 311 VTAB 15: HTAB 1: PRINT CHR$(13)D$"IN#2"
55 312 VTAB 23: HTAB 40: INPUT " ";X,Y,B0
0F 313 IF B0 = 1 OR B0 < 0 THEN 316

```

```
88 314 VTAB CV: HTAB CH: FLASH : PRINT " ";: NORMAL
A8 315 GOTO 312
C2 316 PRINT D$"IN#0"
57 317 IF B0 = 1 THEN IN$ = CHR$ (27): GOTO 319
B0 320 VTAB CV: HTAB CH: PRINT " ";
F7 395 IF CH > 0 THEN HTAB CH
79 396 IF CV > 0 THEN VTAB CV
69 63360 VTAB 1: HTAB LMAX + 5: PRINT D$"IN#2": VTAB
    1: HTAB LMAX + 5: INPUT " ";X0,Y0,B0
61 63370 IF B0 = 1 THEN 63430
88 63380 Y0 = INT (Y0 / 10)
68 63390 VTAB SELECT + 2: HTAB 3: PRINT " ";
C9 63400 SELECT = Y0: IF SELECT > NITEMS THEN SELECT
    = NITEMS
3D 63410 IF SELECT < 1 THEN SELECT = 1
04 63420 GOTO 63350
DC 63430 PRINT D$"IN#0"
```

If you're using ProDOS, change line 311 to the following:

```
01 311 VTAB 15: HTAB 1: PRINT D$"IN#2"
```

The PR#2 and IN#2 in lines 235, 311, and 63360 assume that the mouse interface is in slot 2. If your interface is in another slot, substitute the appropriate slot number in those lines. If you have an Apple IIc, substitute PR#4 and IN#4 for PR#2 and IN#2 in those lines. (Although the IIc doesn't have *physical* slots, the mouse is in *logical* slot 4.) Once you've made all the changes, install the mouse and rerun the program. It works much as described above, using the mouse button instead of Return for menu selections.

Program 1. MOVE Routine

For mistake-proof program entry, use the "Apple MLX" (Appendix C) to type in this program.

START ADDRESS: 9033
END ADDRESS: 91FF

```
9033: AD 59 AA 48 A5 D9 48 A5 26
903B: 76 48 A9 02 85 76 A9 FF 58
9043: 85 D9 A9 BF 85 33 A9 00 1B
904B: 85 F3 4C 56 90 4C 56 4C C9
9053: 56 4C 56 A9 50 85 85 A9 65
905B: 90 A0 00 A2 05 20 FE 90 4E
9063: 4C 68 90 4C 68 A9 66 85 D7
906B: 85 A9 90 A0 00 A2 01 20 82
```



```

9073: FE 90 A9 00 8D 50 90 A9 E5
907B: 04 8D 51 90 AD 51 90 C9 D2
9083: 08 30 0E D0 09 AD 50 90 B3
908B: C9 00 90 05 F0 03 4C EA 0B
9093: 90 AD 50 90 8D A1 90 AD 3D
909B: 51 90 8D A2 90 AD 00 10 B0
90A3: 8D 52 90 A9 00 8D 53 90 3A
90AB: 18 A9 FF 6D 66 90 8D 54 FE
90B3: 90 A9 91 6D 67 90 8D 55 7E
90BB: 90 AD 54 90 8D CC 90 AD 92
90C3: 55 90 8D CD 90 AD 52 90 B2
90CB: 8D 00 10 18 AD 66 90 69 C8
90D3: 01 8D 66 90 AD 67 90 69 44
90DB: 00 8D 67 90 EE 50 90 D0 01
90E3: 03 EE 51 90 4C 7F 90 68 5F
90EB: 85 76 68 85 D9 68 8D 59 B7
90F3: AA A9 8D 8D 01 02 A9 01 C3
90FB: 85 34 60 85 86 84 87 A0 47
9103: 00 A9 00 91 85 C8 D0 02 9C
910B: E6 86 8A D0 04 C6 87 30 1C
9113: 04 CA 4C 04 91 60 AD 59 77
911B: AA 48 A5 D9 48 A5 76 48 06
9123: A9 02 85 76 A9 FF 85 D9 E5
912B: A9 BF 85 33 A9 00 85 F3 43
9133: 4C 3C 91 4C 3C 4C 3C 4C 5A
913B: 3C A9 36 85 85 A9 91 A0 9C
9143: 00 A2 05 20 E4 91 4C 4E 06
914B: 91 4C 4E A9 4C 85 85 A9 DB
9153: 91 A0 00 A2 01 20 E4 91 75
915B: A9 FF 8D 36 91 A9 91 8D 4C
9163: 37 91 AD 37 91 C9 95 30 BE
916B: 0E D0 09 AD 36 91 C9 FF 51
9173: 90 05 F0 03 4C D0 91 AD E4
917B: 36 91 8D 87 91 AD 37 91 8B
9183: 8D 88 91 AD 00 10 8D 38 30
918B: 91 A9 00 8D 39 91 18 A9 A4
9193: 00 6D 4C 91 8D 3A 91 A9 D6
919B: 04 6D 4D 91 8D 3B 91 AD 09
91A3: 3A 91 8D B2 91 AD 3B 91 70
91AB: 8D B3 91 AD 3B 91 8D 00 B2
91B3: 10 18 AD 4C 91 69 01 8D 21
91BB: 4C 91 AD 4D 91 69 00 8D B3
91C3: 4D 91 EE 36 91 D0 03 EE F7
91CB: 37 91 4C 65 91 68 85 76 7E
91D3: 68 85 D9 68 8D 59 AA A9 1F
91DB: 8D 8D 01 02 A9 01 85 34 F9
91E3: 60 85 86 84 87 A0 00 A9 1A
91EB: 00 91 85 C8 D0 02 E6 86 93
91F3: 8A D0 04 C6 87 30 04 CA 4D
91FB: 4C EA 91 60 00 FF FF 00 3B

```

Program 2. Apple II Pull-Down Menus

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```

10 100 REM LINES 150-850 ARE
DB 110 REM A SAMPLE PROGRAM
EA 120 REM DEMONSTRATING
DB 130 REM PULL-DOWN MENUS
BA 140 REM
5B 150 HIMEM: 36914: REM FOR DOS 3.3 ONLY. FOR PRODO
    S USE 35840
14 160 NN = 20: REM    MAXIMUM NUMBER OF ITEMS IN ANY
    MENU
C6 170 DIM MM$(NN): REM MM$=MENU SELECTIONS
62 180 D$ = CHR$(4)
5C 190 PRINT D$"BLOAD MOVE"
FD 200 FOR I = 1 TO 5
BE 210 READ MM$(I)
E2 220 NEXT I
4B 230 HOME
41 240 FOR I = 1 TO 15
57 250 PRINT "THIS IS A SAMPLE PULL-DOWN MENU. ";
EA 260 NEXT I
B3 270 CV = 13:CH = 16
12 280 VTAB 21: HTAB 1: PRINT "-----
-----": REM 39 DASHES
BE 290 PRINT TAB(5)"USE ARROW KEYS TO MOVE CURSOR"
35 300 PRINT TAB(5)"PRESS CTRL-I FOR INSTRUCTIONS"
36 310 PRINT TAB(8)"PRESS ESC KEY FOR MENU ";
BA 320 VTAB CV: HTAB CH
91 330 GET IN$
62 340 IF IN$ = CHR$(9) THEN GOSUB 690
31 350 IF IN$ = CHR$(27) THEN 470
C4 360 IF IN$ = CHR$(8) THEN CH = CH - 1
70 370 IF IN$ = CHR$(21) THEN CH = CH + 1
D9 380 IF IN$ = CHR$(11) THEN CV = CV - 1
53 390 IF IN$ = CHR$(10) THEN CV = CV + 1
BA 400 IF IN$ > CHR$(31) THEN PRINT IN$;:CH = CH +
    1: IF CH > 40 THEN CH = 1:CV = CV + 1
73 410 IF CH < 1 THEN CH = 1
C6 420 IF CH > 40 THEN CH = 40
7E 430 IF CV < 1 THEN CV = 1
4F 440 IF CV > 20 THEN CV = 20
9A 450 GOTO 320
36 460 REM THE FOLLOWING LINE ACTIVATES THE MENU
33 470 TITLE$ = "MENU": GOSUB 63040
59 480 ON SELECT GOTO 280,490,500,510,590

```

```

59 490 HOME : PRINT "THE FIRST FUNCTION OF YOUR PROG
    RAM GOES HERE": GOTO 520
74 500 HOME : PRINT "THE SECOND FUNCTION OF YOUR PRO
    GRAM GOESHERE": GOTO 520
08 510 HOME : PRINT "THE THIRD FUNCTION OF YOUR PROG
    RAM GOES HERE": GOTO 520
3F 520 VTAB 24: PRINT "PRESS ANY KEY TO CONTINUE ...
    ";
D7 530 GET A$
35 540 FOR I = 1 TO NITEMS
97 550 MM$(I) = ""
ED 560 NEXT I
D0 570 RESTORE
1F 580 GOTO 200
25 590 HOME : PRINT "GOOD-BYE!": END
99 600 DATA "LEAVE MENU"
CD 610 DATA "FIRST SELECTION"
71 620 DATA "SECOND SELECTION"
B0 630 DATA "THIRD SELECTION"
AE 640 DATA "QUIT PROGRAM"
98 650 END
93 660 REM
8E 670 REM INSTRUCTIONS
97 680 REM
6C 690 CALL 36915
D5 700 HOME : INVERSE : PRINT BLANK$
DC 710 VTAB 1: HTAB 14: PRINT "INSTRUCTIONS": NORMAL
    : VTAB 3
68 750 PRINT "FOR THIS SAMPLE PROGRAM, YOU CAN MOVE"
12 760 PRINT "THE CURSOR WITH THE ARROW KEYS AND TYP
    E"
39 770 PRINT "ON THE SCREEN. WHEN YOU PRESS ESC, TH
    E"
64 780 PRINT "COMPUTER WILL DISPLAY A PULL DOWN MENU
    ."
30 790 PRINT "USE THE ARROW KEYS TO MOVE THE SELEC-"
47 800 PRINT "TION CURSOR TO THE DESIRED OPTION, THE
    N"
9A 810 PRINT "PRESS RETURN TO SELECT IT."
42 820 VTAB 24: PRINT "PRESS ANY KEY TO CONTINUE ...
    ";
DA 830 GET A$
61 840 CALL 37145
22 850 RETURN
A5 62999 REM #63000
24 63000 REM

```



```

EA 63010 REM  PULL-DOWN MENU
81 63020 REM  SUBROUTINE
3C 63030 REM
90 63040 BLANK$ = "
      ": REM 39 SPACES
A4 63050 LMAX = 0:NITEMS = 0
53 63060 REM DETERMINE MENU SIZE
24 63070 FOR II = 1 TO NN
59 63080 IF MM$(II) = "" THEN 63120
A3 63090 LL = LEN (MM$(II))
62 63100 IF LL > LMAX THEN LMAX = LL
C2 63110 NITEMS = NITEMS + 1
CC 63120 NEXT II
65 63130 IF LMAX > 28 THEN PRINT "NAME IS TOO LONG":
      END
93 63140 REM SAVE SCREEN TEXT
98 63150 CALL 36915
A3 63160 REM DISPLAY MENU
8A 63170 POKE 32,5: POKE 33,LMAX + 5: POKE 34,0: POK
      E 35,NITEMS + 4: REM SET TEXT WINDOW FOR ME
      NU SIZE
6F 63180 HOME
32 63190 INVERSE : PRINT LEFT$ (BL$,LMAX + 5)
D4 63200 VTAB 1: HTAB 3 + ((LMAX - LEN (TITLE$)) / 2
      ): PRINT TITLE$
C0 63210 FOR II = 1 TO NITEMS + 2
0C 63220 VTAB II + 1: HTAB 1: PRINT " ";
6C 63230 HTAB LMAX + 5: PRINT " ";
E0 63240 NEXT II
C0 63250 POKE 35,24
0F 63260 PRINT LEFT$ (BL$,LMAX + 5);
17 63270 POKE 35,NITEMS + 4
D0 63280 VTAB 1
7D 63290 NORMAL
A8 63300 FOR II = 1 TO NITEMS
78 63310 HTAB 4: VTAB II + 2: PRINT MM$(II)
D4 63320 NEXT II
03 63330 REM MAKE SELECTION
99 63340 SELECT = 1
88 63350 HTAB 3: VTAB SELECT + 2: PRINT ">" CHR$ (8)
      ;
43 63360 GET SELECT$
90 63370 HTAB 3: VTAB SELECT + 2: PRINT " "
A4 63380 IF SELECT$ = CHR$ (13) THEN 63480
Df 63390 IF SELECT$ < > CHR$ (10) AND SELECT$ < > CH
      R$ (21) THEN 63430
18 63400 SELECT = SELECT + 1
06 63410 IF SELECT > NITEMS THEN SELECT = 1
04 63420 GOTO 63350

```

```
56 63430 IF SELECT$ < > CHR$ (11) AND SELECT$ < > CH
    R$ (8) GOTO 63350
40 63440 SELECT = SELECT - 1
83 63450 IF SELECT < 1 THEN SELECT = NITEMS
24 63460 GOTO 63350
50 63470 REM RESTORE SCREEN TEXT
83 63480 CALL 37145
94 63490 POKE 32,0: POKE 33,40: POKE 34,0: POKE 35,2
    4: REM RETURN THE TEXT WINDOW TO NORMAL
72 63500 RETURN
```

Mousify Your Applesoft Programs

Lee Swoboda

No Macintosh-style interface would be complete without a mouse. Though you've seen how to add windows (see "Windows") and pull-down menus (see "Apple Pull-Down Menus"), you're still probably using the keyboard for input. This excellent tutorial shows you how to interface and use a mouse with your Apple II computer. (Even though a mouse is preferred, you can use the techniques here with a joystick.) Works with either DOS 3.3 or ProDOS.

Allow me to introduce a new word—*mousify*. Don't try to look that up in a dictionary—it's not there, at least not yet. Though Apple didn't invent the mouse, their Macintosh was the first popular home computer that used one. It's changed the way we "talk" with computers. So the need for the word *mousify*—to add mouse control to computer hardware or software—may grow.

Begin to Mousify

Apple II computers don't come with a mouse, but it's now possible to add one. If you have an Apple II, II+, or IIe, you'll need an interface card (Apple product A2M2050, \$149.95 including the mouse). The interface is built into the IIc, so you need to buy only the mouse device (Apple product A2M4015, \$99.95).

Attaching a mouse to your Apple II is only half the battle. To your computer, a mouse is merely another input device, like a joystick. Many new programs have the ability to accept mouse input. But there are thousands of existing programs that were written before the mouse appeared on the scene. Most commercial programs are written in machine language and protected from unauthorized access, so unless you're an expert machine language programmer, you won't be able to mousify them. However, Applesoft BASIC programs can easily be mousified.

Pages 35–40 of the *AppleMouse II User's Manual*, which comes with the mouse, give a brief description of how to use

the mouse in a BASIC program. But the two examples are trivial, and the text fails to mention some of the "features" that make mousifying an Applesoft program difficult. For example, you must use the INPUT statement to obtain mouse position parameters. Unfortunately, the INPUT statement erases one line of screen text. In addition, your program must set the computer to receive input, which disconnects the keyboard. Mixing mouse and keyboard input requires switching input control between the mouse and the keyboard. Not only that, but you have to use GET statements for keyboard input, which can be exasperating. Using these techniques quickly turns your BASIC program into a Rube Goldberg contraption of INPUTs, GETs, and PRINT D\$s. But there's a better way to handle the mouse.

Fortunately, Apple's input and output (I/O) are *memory-mapped*, meaning that the keyboard and each character on the 40-column screen are at specific locations in Apple's main memory. Applesoft's PEEK and POKE instructions let us examine and change characters on the screen without having to use INPUTs, GETs, or PRINTs. This is especially important when using the mouse, since it lets you zoom quickly to any point on the screen to change a character, rather than performing complex string manipulations.

Practical Application

Let's see how this works. Type in and save Programs 1 and 2; then load and run Program 2, which creates a text file for Program 1 to read.

If your mouse interface is in a slot other than slot 4, change the value assigned to the variable S0 in line 120. If you're using an Apple IIc, for example, the built-in interface is addressed as slot 4, so you would change line 120 to:

```
120 S0 = 4
```

Now load and run Program 1. You'll see an input screen, typical of what might be used in an address book program that lets you store and recall names and addresses. (Of course, this program is just for demonstration—you can't use it to create a real address file.)

In a typical program, the computer would make you reenter an entire line to correct a misspelling or other error. Program 1 lets you point directly at an error with the mouse and

change only the incorrect character. The initial screen display should contain this information:

ENTER INFORMATION

FIRST NAME . COMPUTE!
LAST NAME .. REEDER SERVICE
STREET P.O. BOX 10954
CITY DES MOINES
STATE IA
ZIP 50950
TELEPHONE .. 1-800-247-5470

ERASE QUIT DONE HELP

Notice that the word READER is misspelled REEDER. That's the mistake you'll correct. Also notice that there are three flashing rectangles on the screen. The rapidly flickering rectangle in the upper-right corner is produced when Program 1 obtains mouse input (lines 10150-10160). This effect is always present, but is unimportant to the present discussion. The flashing C at the beginning of the word COMPUTE! is the cursor. The blinking caret (^) in the upper-left corner of the screen is the mouse pointer. The mouse moves the mouse pointer around the screen.

Move the mouse so that the mouse pointer replaces the second E in REEDER and press the mouse button. The computer immediately moves the cursor to the same spot. Now type the letter A (uppercase or lowercase) to correct the spelling mistake. That's all you need to do to correct the error. You can also use the arrow keys to move the cursor (Apple II uses the Ctrl-J and Ctrl-K keys to simulate the up- and down-arrow keys of the Ite and IIC). But the mouse moves the cursor much more rapidly and is far more intuitive to use.

Now move the pointer to the word DONE in the strip menu at the bottom of the screen, and press the button. The computer reads the information from screen memory and, in this case, redisplayes the updated information. Of course, in a working program you would replace lines 30120-30190 with routines that store the data for later recall. You can move the mouse pointer or cursor anywhere on the screen, but line 10710 of the program prevents you from typing anything outside the text area.

How It All Works

Let's take a closer look at the significant portions of Program 1. Lines 130 and 10000-10830 are the most important. Line 130 sets the sensitivity of the mouse. When MI has a value of 20, the pointer moves to any part of the 40-column screen as the mouse moves within a 5×8 inch area. Give MI a larger value to make the mouse less responsive.

Lines 10070-10090 activate the mouse. Input control is transferred from the keyboard to the mouse until line 20030 returns control to the keyboard. Lines 10150-10270 calculate the horizontal and vertical position of the mouse, and determine whether the mouse button has been pushed. Line 10170 and lines 10440-10760 handle input from the keyboard. Note that input control remains with the mouse at this point: The program does *not* use the statement `PRINT D$"IN#0"` to return control to the keyboard. This is the key to the simplicity of Program 1, since it avoids the problems normally encountered when using `GET` and `PRINT` commands with DOS.

Line 10220 and lines 10230-10270 move the cursor to the same position as the mouse pointer when you press the mouse button. Lines 10320-10390 position the mouse pointer and return to line 10150 to read the mouse again. If you don't press a key or the mouse button, the computer stays in the loop from 10150 to 10390, reading the mouse and repositioning the mouse pointer. Lines 10590-10620 position the cursor. This routine is activated only when you press an arrow key or the mouse button. Lines 10640-10690 change all characters to flashing.

Substituting a Joystick

If you don't have a mouse, you can use a joystick to achieve the same effects. With a few modifications, Program 1 can be made to accept joystick input. Here are the steps to follow:

1. Delete lines 120, 130, 10001-10090, and 20220.
2. Modify the following lines as shown:

```

CB 10150 X0 = PDL (0)
04 10160 Y0 = PDL (1)
A0 10170 IF B0 > 127 THEN 10440
A9 10180 Y0 = INT (Y0 / 10) + 1
47 10200 X0 = INT (X0 / 6) + 1
60 10220 IF B0 < 128 THEN 10320
32 20030 REM

```


3. Add the following lines:

```
60 10165 B0 = PEEK ( - 16384)
54 10215 B0 = PEEK ( - 16287)
```

After making these changes, resave Program 1, using a different filename to distinguish it from the original version. When you run it, the joystick moves the mouse pointer around the screen and the button works just like the mouse button. At this point, you might wonder why anyone would buy a mouse, since a joystick seems to work as a substitute. Part of the reason is simply preference—many people find that a real mouse feels better and is therefore more convenient than a joystick. More significantly, most commercial programs that accept mouse input do not recognize input from a joystick. If you're writing programs strictly for your own use, a joystick may serve the purpose; but if you buy commercial software that requires a mouse, you may have no choice.

Using a mouse is a new experience for many Apple II owners. I hope this program inspires you to mousify some of your own programs.

In the next part of this article, we'll expand the capabilities of Program 1 to let you use the mouse to delete and insert blocks of text.

Inserting and Deleting Text

You've just seen how to put a mouse or joystick to work pointing to text on the screen. Now let's take a look at more advanced mouse operations—like defining a text area and deleting, copying, or restoring the defined text. As before, you can substitute a joystick for the mouse if you want.

Functions You Need

Mouse-controlled programs must perform a number of functions in addition to simple pointing. The programs here provide several of these important capabilities:

- **Define text.** Use the mouse to highlight a block of text, which can then be copied or deleted (typical word processing operations).
- **Copy text.** Copy highlighted text to a buffer without deleting it from the screen.
- **Delete text.** Delete highlighted text and save it in a buffer.

- **Insert text.** Restore previously copied or deleted text at a new point on the screen.
- **Cancel.** Undo highlighting if you wish to abort a copy or delete operation.
- **Delete a character.** Delete the character under the cursor.
- **Delete to end of line.** Delete text from the cursor to the end of the line.
- **Find mouse.** Locate the mouse interface.

Getting Started

Enter and save Program 3, which is an expanded and modified version of Program 1. Though it works in either DOS 3.3 or ProDOS, change line 115 if you're using ProDOS:

115 HIMEM: 36352

Before you can run Program 3, you need to create a binary file of the data in Program 4 using the "Apple MLX" machine language entry program from Appendix C. Be sure you read and understand the instructions for MLX before you begin entering the data. Before you load and run MLX (or before you reload MLX if you enter the data in more than one sitting), you must enter the following line in direct mode (without a line number) and press Return:

HIMEM: 36352

Then load and run MLX. You'll be asked for a starting address and an ending address for the data you'll be entering. For this routine, the proper values are as follows:

Starting address: 9200

Ending address: 956F

When you finish entering the data, be sure you save a copy before you leave MLX. For Program 3 to work correctly, you must save the data from Program 4 with the filename **MOUSEY**, and you must save the file on the same disk as Program 3.

Program 5 creates a short text file which we'll use in the following example. If you're using a joystick instead of a mouse, refer to the additional instructions under "Joystick Modifications" below. When you're ready to proceed, your disk should contain a copy of Program 3, a file named **MOUSEY** (created from Program 4), and a file named **TEXT** (created by Program 5).

When you run Program 3, the screen looks like this:

```
ENTER INFORMATION
FIRST NAME . COMPUTE!
LAST NAME .. SUBSCRIPTION SERVICE
ADDRESS .... P.O. BOX 10954
CITY ..... DES MOINES
STATE ..... IA 50340
ZIP .....
TELEPHONE .. 1-800-247-5470

COPY DELETE INSERT CANCEL
ERASE QUIT DONE HELP
```

This screen simulates what you might see in a simple address book program. We've introduced an intentional error by putting the zip code entry on the same line as the state entry. Let's correct the error for a quick demonstration of a few mouse features. Move the mouse cursor to the first number in the zip code, then press and *hold* the mouse button down while moving the mouse to the right. The computer highlights the zip code in reverse video. Keep moving the mouse until all the numbers in the zip code are highlighted, then release the mouse button.

At this point, the highlighted text area has been defined. Now move the mouse pointer to the word **DELETE** in the strip menu at the bottom of the screen and press the mouse button. The computer erases the highlighted zip code from the screen. Don't worry—the information hasn't been lost. Whenever you delete text, the program stores it in a temporary memory buffer.

Now, let's put the zip code data back where it belongs. Move the mouse pointer to the beginning of the next screen line (directly under the *I* in *IA*), then press the mouse button. The computer moves the cursor to that line. Next, move the mouse pointer to the word **INSERT** and press the button again. The zip code data reappears in the desired screen area.

Mouse Editing Functions

Here's a more detailed description of the mouse-editing functions demonstrated in Program 3:

Mouse pointer and text cursor. The rapidly blinking caret (") is the mouse pointer, which you can move around the text screen with the mouse. When the pointer passes over a character, the character blinks rapidly. The flashing rectangle

shows the position of the text cursor. When the cursor passes over a character, the character changes temporarily to flashing uppercase. There are three different ways to move the text cursor:

- Move the mouse pointer to the spot where you want the text cursor to go, then press the mouse button.
- Use the arrow keys as you would in Applesoft BASIC (the Apple II or II+ uses Ctrl-J and Ctrl-K to move up and down, respectively).
- Press Return to move the cursor to the beginning of the next screen line. If the cursor is already on the bottom line, it moves to the top. Pressing Return does not erase the text to the right of the cursor.

Enter text. Text is entered as usual by pressing any letter, number, or punctuation key. Lowercase letters are automatically converted to uppercase.

Define text. Before text can be copied or deleted, you must define it. Move the mouse pointer to the upper-left corner of the text you want to define, then press and hold the mouse button. While pressing the button down, drag the mouse pointer to the lower-right corner of the desired area. The computer marks the defined area by highlighting every character with reverse video. Release the button. The area is defined, and you may proceed to the Cancel, Delete, or Copy options.

Delete text. To delete a text area that you previously defined, move the mouse pointer to DELETE in the strip menu at the bottom of the text screen, then press the button. The computer blanks out the highlighted portion of the screen and stores the first 200 characters of the defined area in a temporary buffer for later use.

Copy text. To copy a text area that you have previously defined, move the pointer to COPY in the strip menu, then press the button. The computer stores the first 200 characters of the defined area in a temporary buffer. Unlike the Delete operation, Copy does not blank out the defined area.

Insert text. To insert text that you previously copied or deleted, move the pointer to the spot where you want to insert text, then press the button to locate the cursor at that spot. Now move the pointer to INSERT in the strip menu and press the button again. The computer inserts the text, using the text

cursor position as a starting point. Note that the inserted text overwrites whatever else was in the affected area. You can insert only the most recently copied or deleted text.

Cancel. If you define a block of text and then decide not to copy or delete it, move the pointer to CANCEL in the strip menu and press the button. The highlighting disappears, and the text is no longer defined.

Editing keys. Press Ctrl-D to delete the character under the cursor. The remaining characters in that line move one space to the left. You can also press Ctrl-X to delete every character from the right of the present cursor position to the end of the line.

Try out the various editing functions. When you've tried everything, move the mouse pointer to DONE in the strip menu and press the button. The demonstration program ends with a routine that reads the current data directly from the video screen.

Since the Copy, Delete, Insert, and Cancel commands are written in BASIC, they may take a second or two to complete if you define a large text area. Though BASIC can't perform such operations very fast, these routines are far easier for you to customize than if they had been written in machine language. If the slowness bothers you, just imagine how long it would take to delete the same amount of text with your trusty pink eraser.

Joystick Modifications

If you don't own a mouse, you can substitute a joystick. Delete lines 120, 130, 10001-10090, 20220, and 44000-44050 from Program 3, then add or modify the lines in Program 6. The joystick moves the mouse pointer around the screen, and the joystick button substitutes for the mouse button.

Since the joystick was designed for a different purpose, its performance doesn't equal that of a mouse. But it costs a lot less.

How It All Works

The machine language routine contained in the MOUSEY file simply highlights text by changing every character between the text cursor and mouse pointer to reverse video. All the other functions are carried out by the BASIC routines in Program 3.

After you define a block of text, lines 35000-44050 act on the highlighted area. The Copy routine (36000-36180) converts each character in the defined area to normal video and stores it in a temporary text buffer. This buffer lies in locations 775-1000 (\$307-\$3E8), a normally unused region.

The Delete routine (37000-37180) is similar to Copy and uses the same temporary buffer, but replaces each character in the defined area with a blank space.

The Insert routine (38000-38100) moves text from the temporary buffer back to the video screen, beginning at the current location of the text cursor.

Lines 39000-40000 contain the Cancel routine, which aborts copy or delete operations. You can also cancel a definition by pressing any key.

The routine at lines 41000-41070 deletes a single character; lines 42000-42060 erase all or part of the current line.

Here are some other useful entry points in the program (note that each of these routines ends with GOTO rather than GOSUB):

Line	Purpose
10120	Reads mouse
10300	Positions mouse pointer
10420	Keyboard input
10570	Positions cursor

Program 1. Apple II Mouse Demonstration

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```

80 120 S0 = 2: REM SLOT CONTAINING MOUSE
07 130 MI = 20: REM MOUSE SENSITIVITY
5A 140 D$ = CHR$ (4)
8C 150 REM
07 160 REM READ DATA FILE
90 170 REM
CB 180 PRINT D$"OPEN TEXT"
32 190 PRINT D$"READ TEXT"
6D 200 INPUT NF$,NL$,AD$,CI$,ST$,ZI$,TE$
CD 210 PRINT D$"CLOSE TEXT"
87 220 REM
25 230 REM DATA ENTRY SCREEN
8B 240 REM
4F 250 HOME
4D 260 Y1 = 4:X1 = 15:C0 = 160
35 270 INVERSE

```



```

D7 280 PRINT "                ENTER INFORMATION
   "
2C 290 VTAB 24: PRINT " MENU:  ERASE  QUIT  DONE  HE
   LP          ";
C6 300 NORMAL
31 310 VTAB 4: HTAB 1
F4 320 PRINT "FIRST NAME .."
C6 330 PRINT "LAST NAME ..."
3C 340 PRINT "STREET ....."
D6 350 PRINT "CITY ....."
1F 360 PRINT "STATE ....."
36 370 PRINT "ZIP ....."
17 380 PRINT "TELEPHONE ..."
3A 390 VTAB 19: HTAB 10: INVERSE : PRINT "^"; NORMA
   L
81 400 PRINT " IS MOUSE POINTER"
3C 410 VTAB 21: HTAB 14: INVERSE : PRINT " "; NORMA
   L
38 420 PRINT " IS CURSOR"
26 430 VTAB 4
5E 440 HTAB 15: PRINT NF$
66 450 HTAB 15: PRINT NL$
D9 460 HTAB 15: PRINT AD$
E1 470 HTAB 15: PRINT CI$
F6 480 HTAB 15: PRINT ST$
71 490 HTAB 15: PRINT ZI$
59 500 HTAB 15: PRINT TE$
73 9999 REM #100000
19 10000 REM
E6 10001 REM -----
29 10010 REM MOUSE ROUTINES
E6 10020 REM -----
39 10040 REM
A4 10050 REM TURN MOUSE "ON"
49 10060 REM
A8 10070 PRINT D$"PR#"S0: PRINT CHR$ (1)
CB 10080 PRINT D$"PR#0"
09 10090 PRINT D$"IN#"S0
17 10100 GOTO 10590
25 10110 REM
65 10120 REM DETERMINE POSITION
91 10130 REM OF MOUSE
3D 10140 REM
1C 10150 VTAB 1: HTAB 40
77 10160 INPUT " "; X0,Y0,B0
7D 10170 IF B0 < 0 THEN 10440: REM KEY PRESSED?
D0 10180 Y0 = INT (Y0 / MI) + 1
78 10190 IF Y0 > 24 THEN Y0 = 24
64 10200 X0 = INT (X0 / MI) + 1
75 10210 IF X0 > 40 THEN X0 = 40

```

```

0B 10220 IF B0 > 1 THEN 10320: REM BUTTON PRESSED?
B9 10230 IF Y0 = 24 THEN 20030
63 10240 Y1 = Y0: X1 = X0
7B 10250 POKE V0, C0
4B 10260 C0 = C2
F2 10270 GOSUB 10800
F2 10280 GOTO 10620
69 10290 REM
ED 10300 REM POSITION MOUSE POINTER
2D 10310 REM
86 10320 IF V0 = V1 THEN C2 = C1
B0 10330 POKE V1, C2
A2 10340 V1 = 1023 + 128 * (Y0 - 1) + X0
3F 10350 IF Y0 > 8 THEN V1 = V1 - 984
9C 10360 IF Y0 > 16 THEN V1 = V1 - 984
27 10370 C2 = PEEK (V1)
64 10380 POKE V1, 160
0A 10390 IF C2 = 160 THEN POKE V1, 30
C2 10400 GOTO 10150
31 10410 REM
01 10420 REM KEYBOARD INPUT
41 10430 REM
F9 10440 C3 = PEEK ( - 16384)
77 10450 POKE - 16368, 0
DC 10455 IF C3 > 223 THEN C3 = C3 - 32: REM CONVERT
      TO UPPER CASE
48 10460 IF C3 > 159 THEN 10710
CD 10470 IF C3 = 141 THEN X1 = 15: Y1 = Y1 + 1: IF Y1
      > 10 THEN Y1 = 4: REM RETURN KEY
69 10480 IF C3 = 139 THEN Y1 = Y1 + 1: REM DOWN ARRO
      W
B2 10490 IF C3 = 138 THEN Y1 = Y1 - 1: REM UP ARROW
BF 10500 IF C3 = 149 THEN X1 = X1 + 1: REM RIGHT ARR
      OW
71 10510 IF C3 = 136 THEN X1 = X1 - 1: REM LEFT ARRO
      W
56 10520 IF Y1 > 24 THEN Y1 = 24
DC 10530 IF Y1 < 1 THEN Y1 = 1
9C 10540 IF X1 > 40 THEN X1 = 40
EB 10550 IF X1 < 1 THEN X1 = 1
5D 10560 REM
06 10570 REM POSITION CURSOR
6D 10580 REM
A4 10590 POKE V0, C0
CA 10600 GOSUB 10800
42 10610 C0 = PEEK (V0)
9E 10620 IF V0 = V1 THEN C0 = C2
44 10630 REM CHANGE TO FLASHING CHARACTER
87 10640 C1 = C0
23 10650 IF C1 > 127 THEN C1 = C1 - 64

```

```

7F 10660 IF C1 > 64 THEN C1 = C1 - 64
D9 10670 IF C1 > 95 THEN C1 = C1 - 32
4B 10680 IF C1 < 64 THEN C1 = C1 + 64
C8 10690 POKE V0,C1
CE 10700 GOTO 10150
6B 10710 IF X1 < 15 OR Y1 < 4 OR Y1 > 10 THEN 10150
DE 10720 GOSUB 10800
DC 10730 POKE V0,C3
51 10740 C0 = C3
CE 10750 IF V0 = V1 THEN C2 = C3
0C 10760 X1 = X1 + 1: IF X1 > 39 THEN X1 = 39
67 10770 GOTO 10590
16 10780 REM CALCULATE V0
6E 10790 REM (VIDEO BUFFER ADDRESS)
60 10800 V0 = 1023 + 128 * (Y1 - 1) + X1
2B 10810 IF Y1 > 8 THEN V0 = V0 - 984
7F 10820 IF Y1 > 16 THEN V0 = V0 - 984
8B 10830 RETURN
9A 19999 REM #20000
1A 20000 REM
AE 20010 REM STRIP MENU
2A 20020 REM
C2 20030 PRINT D$"IN#0"
CB 20040 IF X0 > 8 AND X0 < 14 THEN NF$ = "":NL$ = "
      ":AD$ = "":CI$ = "":ST$ = "":ZI$ = "":TE$ =
      "": GOTO 250
1F 20050 IF X0 > 15 AND X0 < 20 THEN HOME : END
73 20060 IF X0 > 21 AND X0 < 26 THEN 30030
7A 20070 IF X0 > 27 AND X0 < 32 THEN 20100
71 20080 VTAB 1: HTAB 40: PRINT D$"IN#"S0: GOTO 1015
      0
17 20090 REM HELP TEXT
CD 20100 VTAB 12: HTAB 1
8A 20110 PRINT "THE FLASHING REFLEX (^) IS THE MOUSE
      "
75 20120 PRINT "POINTER AND THE FLASHING RECTANGLE I
      S"
4B 20130 PRINT "THE CURSOR. TO MOVE THE CURSOR TO T
      HE"
36 20140 PRINT "ENTRY YOU WANT TO CHANGE, USE THE AR
      ROW"
4E 20150 PRINT "KEYS OR USE THE MOUSE TO MOVE THE MO
      USE"
47 20160 PRINT "POINTER, THEN PRESS THE MOUSE BUTTON
      TO"
E6 20170 PRINT "MOVE THE CURSOR TO THAT POINT. TYPE
      "
EA 20180 PRINT "NEW OR CORRECTED DATA, THEN MOVE THE
      "
31 20190 PRINT "MOUSE CURSOR TO 'DONE' IN THE MENU"

```



```

4A 20200 PRINT "BELOW AND PRESS THE MOUSE BUTTON TO"
04 20210 PRINT "ACCEPT THE ENTRIES ABOVE."
29 20220 PRINT D$"IN#"S0
03 20230 GOTO 10150
90 29999 REM #30000
18 30000 REM
28 30010 REM EXAMPLE
28 30020 REM
A1 30030 Y1 = 4: GOSUB 63050:NF$ = A$
2C 30040 Y1 = 5: GOSUB 63050:NL$ = A$
91 30050 Y1 = 6: GOSUB 63050:AD$ = A$
1C 30060 Y1 = 7: GOSUB 63050:CI$ = A$
E9 30070 Y1 = 8: GOSUB 63050:ST$ = A$
11 30080 Y1 = 9: GOSUB 63050:ZI$ = A$
17 30090 Y1 = 10: GOSUB 63050:TE$ = A$
2E 30100 REM GO TO REMAINDER OF YOUR PROGRAM
1C 30110 REM FOR EXAMPLE ...
36 30120 HOME
5E 30130 VTAB 10
5E 30140 PRINT NF$" "NL$
38 30150 PRINT AD$
90 30160 PRINT CI$", "ST$" "ZI$
9C 30170 PRINT TE$
CA 30180 CALL - 198: CALL - 198
89 30190 END : REM END OF EXAMPLE
A5 62999 REM #63000
24 63000 REM
2C 63010 REM SUBROUTINE TO "READ"
1F 63020 REM STRINGS FROM THE
B3 63030 REM VIDEO BUFFER
44 63040 REM
0F 63050 VTAB 24: FLASH : PRINT " WORKING ...
";: NORMAL : VTAB 1: HT

AB 1
C9 63060 A$ = ""
FC 63070 REM CALCULATE V0
55 63080 REM (VIDEO BUFFER ADDRESS)
A5 63090 V0 = 1037 + 128 * (Y1 - 1)
12 63100 IF Y1 > 8 THEN V0 = V0 - 984
66 63110 IF Y1 > 16 THEN V0 = V0 - 984
2F 63120 FOR I = 1 TO 25
67 63130 C0 = PEEK (V0 + I)
00 63140 IF C0 = 160 AND PEEK (V0 + I + 1) = 160 THEN
N 63190: REM END IF TWO BLANKS
F9 63160 IF C0 > 128 THEN C0 = C0 - 128
F5 63170 A$ = A$ + CHR$ (C0)
05 63180 NEXT I
C2 63190 IF RIGHT$ (A$,1) = CHR$ (32) THEN A$ = LEFT
$ (A$, LEN (A$) - 1): GOTO 63190: REM REMOV
E TRAILING BLANKS
66 63200 RETURN

```

Program 2. Sample Screen Maker

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
51 10 D$ = CHR$ (4)
07 20 PRINT D$"OPEN TEXT"
CF 30 PRINT D$"WRITE TEXT"
EA 40 PRINT "COMPUTE!"
8E 50 PRINT "REEDER SERVICE"
D1 60 PRINT "P.O. BOX 10954"
E3 70 PRINT "DES MOINES"
C8 80 PRINT "IA"
FC 90 PRINT "50950"
91 100 PRINT "1-800-247-5470"
CC 110 PRINT D$"CLOSE TEXT"
```

Program 3. Advanced Mousification

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
20 115 HIMEM: 37375
DF 120 GOSUB 44010
D7 130 MI = 20: REM MOUSE SENSITIVITY
5A 140 D$ = CHR$ (4)
E8 145 PRINT D$"BLOAD MOUSEY"
8C 150 REM
07 160 REM READ DATA FILE
90 170 REM
76 180 PRINT D$"OPEN TEXT2"
DC 190 PRINT D$"READ TEXT2"
6D 200 INPUT NF$,NL$,AD$,CI$,ST$,ZI$,TE$
23 210 PRINT D$"CLOSE TEXT2"
87 220 REM
25 230 REM DATA ENTRY SCREEN
9B 240 REM
4F 250 HOME
4D 260 Y1 = 4: X1 = 15: C0 = 160
35 270 INVERSE
D7 280 PRINT "                ENTER INFORMATION
    "
24 285 VTAB 23: PRINT "                COPY DELETE INSERT C
    ANCEL                "
9B 290 VTAB 24: PRINT "                ERASE QUIT DONE HE
    LP                ";
C6 300 NORMAL
31 310 VTAB 4: HTAB 1
F4 320 PRINT "FIRST NAME .."
C6 330 PRINT "LAST NAME ..."
3C 340 PRINT "STREET ....."
```

```

D6 350 PRINT "CITY ....."
1F 360 PRINT "STATE ....."
36 370 PRINT "ZIP ....."
17 380 PRINT "TELEPHONE ..."
3A 390 VTAB 19: HTAB 10: INVERSE : PRINT "^";: NORMA
    L
91 400 PRINT " IS MOUSE POINTER"
3C 410 VTAB 21: HTAB 14: INVERSE : PRINT " ";: NORMA
    L
38 420 PRINT " IS CURSOR"
26 430 VTAB 4
5E 440 HTAB 15: PRINT NF$
66 450 HTAB 15: PRINT NL$
D9 460 HTAB 15: PRINT AD$
E1 470 HTAB 15: PRINT CI$
F6 480 HTAB 15: PRINT ST$
71 490 HTAB 15: PRINT ZI$
59 500 HTAB 15: PRINT TE$
73 9999 REM #10000
19 10000 REM
E6 10001 REM -----
29 10010 REM MOUSE ROUTINES
E6 10020 REM -----
39 10040 REM
A4 10050 REM TURN MOUSE "ON"
49 10060 REM
A8 10070 PRINT D$"PR#"S0: PRINT CHR$ (1)
CB 10080 PRINT D$"PR#0"
09 10090 PRINT D$"IN#"S0
17 10100 GOTO 10590
25 10110 REM
65 10120 REM DETERMINE POSITION
91 10130 REM OF MOUSE
3D 10140 REM
1C 10150 VTAB 1: HTAB 40
77 10160 INPUT " ";X0,Y0,B0
7D 10170 IF B0 < 0 THEN 10440: REM KEY PRESSED?
D0 10180 Y0 = INT (Y0 / MI) + 1
7B 10190 IF Y0 > 24 THEN Y0 = 24
64 10200 X0 = INT (X0 / MI) + 1
75 10210 IF X0 > 40 THEN X0 = 40
6A 10215 IF B0 = 2 AND X0 > 20 AND X0 < 27 AND Y0 =
    23 THEN 38010
6D 10216 IF B0 = 2 AND SW = 0 THEN X2 = X0:Y2 = Y0:
    POKE 768,Y2: POKE 769,X2:X3 = X0:Y3 = Y0
9B 10217 IF B0 = 3 AND SW = 2 THEN SW = 3
31 10218 ON SW GOTO 35010,35020,10315
0B 10220 IF B0 > 1 THEN 10320: REM BUTTON PRESSED?
D9 10225 IF X2 < > X0 OR Y2 < > Y0 THEN SW = 1: GOTO
    35010

```



```
B9 10230 IF Y0 = 24 THEN 20030
63 10240 Y1 = Y0:X1 = X0
78 10250 POKE V0,C0
E0 10255 IF C0 < 128 THEN POKE V0,C0 + 128
48 10260 C0 = C2
F2 10270 GOSUB 10800
F2 10280 GOTO 10620
69 10290 REM
ED 10300 REM POSITION MOUSE POINTER
2D 10310 REM
DD 10315 IF B0 = 2 AND Y0 = 23 THEN 20081
86 10320 IF V0 = V1 THEN C2 = C1
B0 10330 POKE V1,C2
A2 10340 V1 = 1023 + 128 * (Y0 - 1) + X0
3F 10350 IF Y0 > 8 THEN V1 = V1 - 984
9C 10360 IF Y0 > 16 THEN V1 = V1 - 984
27 10370 C2 = PEEK (V1)
64 10380 POKE V1,160
0A 10390 IF C2 = 160 THEN POKE V1,30
C2 10400 GOTO 10150
31 10410 REM
01 10420 REM KEYBOARD INPUT
41 10430 REM
F9 10440 C3 = PEEK ( - 16384)
77 10450 POKE - 16368,0
DC 10455 IF C3 > 223 THEN C3 = C3 - 32: REM CONVERT
    TO UPPER CASE
7C 10456 IF SW > 0 THEN GOSUB 39010
48 10460 IF C3 > 159 THEN 10710
63 10465 IF C3 = 132 OR C3 = 225 THEN IF X1 > 14 AND
    Y1 > 3 AND Y1 < 11 THEN GOSUB 41010
0A 10466 IF C3 = 152 THEN IF X1 > 14 AND Y1 > 3 AND
    Y1 < 11 THEN GOSUB 42010
CD 10470 IF C3 = 141 THEN X1 = 15:Y1 = Y1 + 1: IF Y1
    > 10 THEN Y1 = 4: REM RETURN KEY
59 10480 IF C3 = 138 THEN Y1 = Y1 + 1: REM DOWN ARRO
    W
C2 10490 IF C3 = 139 THEN Y1 = Y1 - 1: REM UP ARROW
BF 10500 IF C3 = 149 THEN X1 = X1 + 1: REM RIGHT ARR
    OW
71 10510 IF C3 = 136 THEN X1 = X1 - 1: REM LEFT ARRO
    W
56 10520 IF Y1 > 24 THEN Y1 = 24
DC 10530 IF Y1 < 1 THEN Y1 = 1
9C 10540 IF X1 > 40 THEN X1 = 40
EB 10550 IF X1 < 1 THEN X1 = 1
5D 10560 REM
06 10570 REM POSITION CURSOR
6D 10580 REM
A4 10590 POKE V0,C0
```

```

CA 10600 GOSUB 10800
42 10610 C0 = PEEK (V0)
9E 10620 IF V0 = V1 THEN C0 = C2
44 10630 REM CHANGE TO FLASHING CHARACTER
87 10640 C1 = C0
23 10650 IF C1 > 127 THEN C1 = C1 - 64
7F 10660 IF C1 > 64 THEN C1 = C1 - 64
D9 10670 IF C1 > 95 THEN C1 = C1 - 32
4B 10680 IF C1 < 64 THEN C1 = C1 + 64
C8 10690 POKE V0,C1
CE 10700 GOTO 10150
6B 10710 IF X1 < 15 OR Y1 < 4 OR Y1 > 10 THEN 10150
DE 10720 GOSUB 10800
DC 10730 POKE V0,C3
51 10740 C0 = C3
CE 10750 IF V0 = V1 THEN C2 = C3
0C 10760 X1 = X1 + 1: IF X1 > 39 THEN X1 = 39
67 10770 GOTO 10590
16 10780 REM CALCULATE V0
6E 10790 REM (VIDEO BUFFER ADDRESS)
60 10800 V0 = 1023 + 128 * (Y1 - 1) + X1
2B 10810 IF Y1 > 8 THEN V0 = V0 - 984
7F 10820 IF Y1 > 16 THEN V0 = V0 - 984
8B 10830 RETURN
9A 19999 REM #200000
1A 20000 REM
AE 20010 REM STRIP MENU
2A 20020 REM
C2 20030 PRINT D$"IN#0"
CB 20040 IF X0 > 8 AND X0 < 14 THEN NF$ = "":NL$ = "
      ":AD$ = "":CI$ = "":ST$ = "":ZI$ = "":TE$ =
      "": GOTO 250
1F 20050 IF X0 > 15 AND X0 < 20 THEN HOME : END
73 20060 IF X0 > 21 AND X0 < 26 THEN 30030
7A 20070 IF X0 > 27 AND X0 < 32 THEN 20100
71 20080 VTAB 1: HTAB 40: PRINT D$"IN#"S0: GOTO 1015
      0
5C 20081 IF X0 > 6 AND X0 < 11 THEN GOSUB 36010: GOT
      O 10590
82 20082 IF X0 > 12 AND X0 < 19 THEN GOSUB 37010: GO
      TO 10590
5F 20083 IF X0 > 28 AND X0 < 35 THEN GOSUB 39010: GO
      TO 10590
34 20084 GOTO 10150
17 20090 REM HELP TEXT
CD 20100 VTAB 12: HTAB 1
8A 20110 PRINT "THE FLASHING REFLEX (^) IS THE MOUSE
      "
75 20120 PRINT "POINTER AND THE FLASHING RECTANGLE I
      S"

```

```
4B 20130 PRINT "THE CURSOR.  TO MOVE THE CURSOR TO T
    HE"
36 20140 PRINT "ENTRY YOU WANT TO CHANGE, USE THE AR
    ROW"
4E 20150 PRINT "KEYS OR USE THE MOUSE TO MOVE THE MO
    USE"
47 20160 PRINT "POINTER, THEN PRESS THE MOUSE BUTTON
    TO"
E6 20170 PRINT "MOVE THE CURSOR TO THAT POINT.  TYPE
    "
EA 20180 PRINT "NEW OR CORRECTED DATA, THEN MOVE THE
    "
31 20190 PRINT "MOUSE CURSOR TO 'DONE' IN THE MENU"
4A 20200 PRINT "BELOW AND PRESS THE MOUSE BUTTON TO"
D4 20210 PRINT "ACCEPT THE ENTRIES ABOVE."
D9 20220 PRINT D$"IN#"S0
D3 20230 GOTO 10150
9D 29999 REM #30000
1B 30000 REM
2B 30010 REM EXAMPLE
2B 30020 REM
A1 30030 Y1 = 4: GOSUB 63050:NF$ = A$
2C 30040 Y1 = 5: GOSUB 63050:NL$ = A$
91 30050 Y1 = 6: GOSUB 63050:AD$ = A$
1C 30060 Y1 = 7: GOSUB 63050:CI$ = A$
E9 30070 Y1 = 8: GOSUB 63050:ST$ = A$
11 30080 Y1 = 9: GOSUB 63050:ZI$ = A$
17 30090 Y1 = 10: GOSUB 63050:TE$ = A$
2E 30100 REM GO TO REMAINDER OF YOUR PROGRAM
1C 30110 REM FOR EXAMPLE ...
36 30120 HOME
5E 30130 VTAB 10
EE 30140 PRINT NF$" "NL$
3B 30150 PRINT AD$
B0 30160 PRINT CI$", "ST$" "ZI$
9C 30170 PRINT TE$
CA 30180 CALL - 198: CALL - 198
89 30190 END : REM END OF EXAMPLE
EC 35000 REM HIGHLIGHT TEXT
AD 35010 POKE V0,C1 - 64:SW = 2
AC 35020 IF B0 > 1 THEN 10150
40 35030 IF X3 < X0 THEN X3 = X0
D5 35040 IF Y3 < Y0 THEN Y3 = Y0
BB 35050 POKE 770,Y3: POKE 771,X3
8D 35060 POKE 772,Y0: POKE 773,X0
FC 35070 CALL 37376
2B 35080 Y3 = Y0:X3 = X0
07 35090 GOTO 10150
9C 36000 REM COPY
60 36010 P3 = 775
```



```

38 36020 FOR II = Y2 TO Y3
BA 36030 FOR JJ = X2 TO X3
AC 36040 GOSUB 40020
39 36050 C3 = PEEK (V2) + 128
96 36060 IF C3 < 160 THEN C3 = C3 + 64
37 36070 IF C3 > 223 THEN C3 = C3 - 64
FE 36080 POKE V2,C3
CD 36090 IF P3 > 1000 THEN 36120
BA 36100 POKE P3,C3
DA 36110 P3 = P3 + 1
D5 36120 NEXT JJ
3E 36130 POKE P3,141
EE 36140 IF P3 < 1001 THEN P3 = P3 + 1
E7 36150 NEXT II
87 36160 POKE P3,255
B0 36170 SW = 0
A5 36180 RETURN
E0 37000 REM DELETE
62 37010 P3 = 775
3A 37020 FOR II = Y2 TO Y3
9C 37030 FOR JJ = X2 TO X3
AE 37040 GOSUB 40020
3B 37050 C3 = PEEK (V2) + 128
98 37060 IF C3 < 160 THEN C3 = C3 + 64
39 37070 IF C3 > 223 THEN C3 = C3 - 64
6C 37080 POKE V2,160
D7 37090 IF P3 > 1000 THEN 37120
BC 37100 POKE P3,C3
DC 37110 P3 = P3 + 1
D7 37120 NEXT JJ
40 37130 POKE P3,141
F0 37140 IF P3 < 1001 THEN P3 = P3 + 1
E9 37150 NEXT II
89 37160 POKE P3,255
D4 37170 SW = 0:C0 = 160
A7 37180 RETURN
D7 38000 REM INSERT
64 38010 P3 = 775
05 38020 II = Y1:JJ = X1
0D 38030 C3 = PEEK (P3)
F1 38040 IF C3 = 141 THEN II = II + 1:JJ = X1:P3 = P
    3 + 1: GOTO 38030
72 38050 IF II > 22 OR JJ > 40 THEN 38090
D0 38060 IF C3 = 255 THEN SW = 0:C0 = PEEK (V0): GOT
    0 10590
C8 38070 GOSUB 40020
03 38080 POKE V2,C3
4E 38090 JJ = JJ + 1:P3 = P3 + 1
C4 38100 GOTO 38030
C3 39000 REM CANCEL

```

```
36 39010 FOR II = Y2 TO Y3
88 39020 FOR JJ = X2 TO X3
AA 39030 GOSUB 40020
37 39040 C3 = PEEK (V2) + 128
94 39050 IF C3 < 160 THEN C3 = C3 + 64
35 39060 IF C3 > 223 THEN C3 = C3 - 64
FC 39070 POKE V2,C3
43 39080 NEXT JJ,II
C2 39090 SW = 0
56 40000 RETURN
24 40010 REM
78 40020 V2 = 1023 + 128 * (II - 1) + JJ
40 40030 IF II > 8 THEN V2 = V2 - 984
A6 40040 IF II > 16 THEN V2 = V2 - 984
7E 40050 RETURN
7F 41000 REM DELETE A CHARACTER
83 41010 GOSUB 43010
96 41020 FOR II = V0 TO V2 - 1
8B 41030 POKE II, PEEK (II + 1)
D2 41040 NEXT II
49 41050 POKE V2,160
57 41060 C0 = PEEK (V0)
90 41070 RETURN
8F 42000 REM DELETE TO END OF LINE
85 42010 GOSUB 43010
76 42020 FOR II = V0 TO V2
7D 42030 POKE II,160
D4 42040 NEXT II
FA 42050 C0 = 160
8A 42060 RETURN
22 43000 REM
CF 43010 V2 = 1063 + 128 * (Y1 - 1)
2E 43020 IF Y1 > 8 THEN V2 = V2 - 984
94 43030 IF Y1 > 16 THEN V2 = V2 - 984
7C 43040 RETURN
DB 44000 REM FIND MOUSE
56 44010 FOR S0 = 0 TO 6
AA 44020 IF PEEK (49420 + (256 * S0)) = 32 AND PEEK
      (49659 + (256 * S0)) = 214 THEN S0 = S0 + 1
      : RETURN
80 44030 NEXT S0
82 44040 PRINT "I CAN'T FIND A MOUSE INTERFACE CARD"
      CHR$ (7) CHR$ (7)
68 44050 END
A5 62999 REM #63000
24 63000 REM
2C 63010 REM SUBROUTINE TO "READ"
IF 63020 REM STRINGS FROM THE
83 63030 REM VIDEO BUFFER
44 63040 REM
```

```

0F 63050 VTAB 24: FLASH : PRINT "  WORKING ...
                                ";: NORMAL : VTAB 1: HT
      AB 1
C9 63060 A$ = ""
FC 63070 REM CALCULATE V0
55 63080 REM (VIDEO BUFFER ADDRESS)
A5 63090 V0 = 1037 + 128 * (Y1 - 1)
12 63100 IF Y1 > 8 THEN V0 = V0 - 984
66 63110 IF Y1 > 16 THEN V0 = V0 - 984
2F 63120 FOR I = 1 TO 25
67 63130 C0 = PEEK (V0 + I)
AD 63135 IF C0 < 128 THEN C0 = C0 + 128
70 63136 IF C0 < 160 THEN C0 = C0 + 64
19 63137 IF C0 > 223 THEN C0 = C0 - 64
DD 63140 IF C0 = 160 AND PEEK (V0 + I + 1) = 160 THE
      N 63190: REM END IF TWO BLANKS
F9 63160 IF C0 > 128 THEN C0 = C0 - 128
F5 63170 A$ = A$ + CHR$ (C0)
D5 63180 NEXT I
C2 63190 IF RIGHT$ (A$,1) = CHR$ (32) THEN A$ = LEFT
      $ (A$, LEN (A$) - 1): GOTO 63190: REM REMOV
      E TRAILING BLANKS
42 63195 IF A$ = CHR$ (96) THEN A$ = ""
66 63200 RETURN

```

Program 4. MOUSEY File

For mistake-proof program entry, use the "Apple MLX" (Appendix C) to type in this program.

START ADDRESS: 9200
END ADDRESS: 956F

```

9200: AD 59 AA 48 A5 D9 48 A5 F6
9208: 76 48 A9 02 85 76 A9 FF 29
9210: 85 D9 A9 BF 85 33 A9 00 EB
9218: 85 F3 4C 29 92 04 00 0F CC
9220: 00 07 00 15 00 07 00 15 89
9228: 00 A9 1D 85 85 A9 92 A0 4C
9230: 00 A2 0B 20 4D 95 4C 45 FF
9238: 92 14 07 20 00 08 00 16 C4
9240: 00 08 00 16 00 A9 39 85 67
9248: 85 A9 92 A0 00 A2 0B 20 B7
9250: 4D 95 AD 00 03 8D 1D 92 52
9258: A9 00 8D 1E 92 AD 01 03 36
9260: 8D 1F 92 A9 00 8D 20 92 0A
9268: AD 02 03 8D 21 92 A9 00 C4
9270: 8D 22 92 AD 03 03 8D 23 74
9278: 92 A9 00 8D 24 92 AD 04 F4
9280: 03 8D 25 92 A9 00 8D 26 E6

```


9288: 92 AD 05 03 8D 27 92 A9 0B
9290: 00 8D 28 92 AD 1D 92 8D DB
9298: 3D 92 AD 1E 92 8D 3E 92 72
92A0: AD 3E 92 CD 22 92 30 0F 26
92A8: D0 0A AD 3D 92 CD 21 92 E2
92B0: 90 05 F0 03 4C 86 93 AD FE
92B8: 1F 92 8D 3F 92 AD 20 92 D5
92C0: 8D 40 92 AD 40 92 CD 24 F5
92C8: 92 30 0F D0 0A AD 3F 92 4A
92D0: CD 23 92 90 05 F0 03 4C 3F
92D8: 7B 93 18 AD 25 92 69 01 C5
92E0: 8D 41 92 AD 26 92 69 00 98
92E8: 8D 42 92 18 AD 27 92 69 D1
92F0: 01 8D 43 92 AD 28 92 69 28
92F8: 00 8D 44 92 AD 3E 92 CD 8C
9300: 42 92 30 0A D0 1D AD 3D 27
9308: 92 CD 41 92 B0 15 AD 40 B2
9310: 92 CD 44 92 30 0A D0 0B FB
9318: AD 3F 92 CD 43 92 B0 03 DD
9320: 4C 70 93 20 7A 94 AD 39 B8
9328: 92 8D 33 93 AD 3A 92 8D A4
9330: 34 93 AD 90 07 8D 3B 92 8C
9338: A9 00 8D 3C 92 AD 3C 92 FF
9340: C9 00 30 09 D0 18 AD 3B 60
9348: 92 C9 80 B0 11 18 AD 3B C5
9350: 92 69 80 8D 3B 92 AD 3C BF
9358: 92 69 00 8D 3C 92 AD 39 BC
9360: 92 8D 6E 93 AD 3A 92 8D 44
9368: 6F 93 AD 3B 92 8D 90 07 88
9370: EE 3F 92 D0 03 EE 40 92 25
9378: 4C C3 92 EE 3D 92 D0 03 D0
9380: EE 3E 92 4C A0 92 AD 1D 8D
9388: 92 8D 3D 92 AD 1E 92 8D C5
9390: 3E 92 AD 3E 92 CD 26 92 BF
9398: 30 0F D0 0A AD 3D 92 CD AB
93A0: 25 92 90 05 F0 03 4C 77 05
93A8: 94 AD 1F 92 8D 3F 92 AD CE
93B0: 20 92 8D 40 92 AD 40 92 A0
93B8: CD 28 92 30 0F D0 0A AD A3
93C0: 3F 92 CD 27 92 90 05 F0 2A
93C8: 03 4C 6C 94 20 7A 94 AD 1D
93D0: 39 92 8D DC 93 AD 3A 92 13
93D8: 8D DD 93 AD 14 07 8D 3B 9E
93E0: 92 A9 00 8D 3C 92 AD 3C 58
93E8: 92 C9 00 30 1C D0 09 AD B2
93F0: 3B 92 C9 7F 90 13 F0 11 4F
93F8: 38 AD 3B 92 E9 40 8D 3B DE
9400: 92 AD 3C 92 E9 00 8D 3C 35
9408: 92 AD 3C 92 C9 00 30 1C 61
9410: D0 09 AD 3B 92 C9 40 90 1A

```

9418: 13 F0 11 38 AD 3B 92 E9 16
9420: 40 8D 3B 92 AD 3C 92 E9 CA
9428: 00 8D 3C 92 AD 3C 92 C9 B2
9430: 00 30 1C D0 09 AD 3B 92 FD
9438: C9 40 90 13 F0 11 38 AD 83
9440: 3B 92 E9 40 8D 3B 92 AD 19
9448: 3C 92 E9 00 8D 3C 92 AD A1
9450: 39 92 8D 5F 94 AD 3A 92 C4
9458: 8D 60 94 AD 3B 92 8D 14 21
9460: 07 EE 3F 92 D0 03 EE 40 8A
9468: 92 4C B5 93 EE 3D 92 D0 40
9470: 03 EE 3E 92 4C 92 93 4C E7
9478: 0B 95 38 AD 3D 92 E9 01 77
9480: 8D 39 92 AD 3E 92 E9 00 FB
9488: 8D 3A 92 A9 00 85 8A A9 CB
9490: 80 AE 3A 92 AC 39 92 20 A5
9498: 1F 95 8E 3A 92 8C 39 92 F7
94A0: 18 A9 FF 6D 39 92 8D 39 7F
94A8: 92 A9 03 6D 3A 92 8D 3A 2E
94B0: 92 18 AD 39 92 6D 3F 92 CD
94B8: 8D 39 92 AD 3A 92 6D 40 5B
94C0: 92 8D 3A 92 AD 3E 92 C9 5C
94C8: 00 30 1C D0 09 AD 3D 92 9A
94D0: C9 08 90 13 F0 11 38 AD 0E
94D8: 39 92 E9 D8 8D 39 92 AD 32
94E0: 3A 92 E9 03 8D 3A 92 AD 61
94E8: 3E 92 C9 00 30 1C D0 09 AB
94F0: AD 3D 92 C9 10 90 13 F0 09
94F8: 11 38 AD 39 92 E9 D8 8D 7D
9500: 39 92 AD 3A 92 E9 03 8D 95
9508: 3A 92 60 68 85 76 68 85 E3
9510: D9 68 8D 59 AA A9 8D 8D 2E
9518: 01 02 A9 01 85 34 60 85 CC
9520: 89 84 87 86 88 A9 00 85 FA
9528: 85 85 86 46 88 66 87 90 2A
9530: 0D 18 A5 89 65 85 85 85 07
9538: A5 8A 65 86 85 86 06 89 C9
9540: 26 8A A5 88 05 87 D0 E3 2A
9548: A4 85 A6 86 60 85 86 84 0F
9550: 87 A0 00 A9 00 91 85 C8 1C
9558: D0 02 E6 86 8A D0 04 C6 18
9560: 87 30 04 CA 4C 53 95 60 C3
9568: 00 FF 00 00 FF FF 00 00 93

```

Program 5. TEXT Filemaker

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
51 10 D$ = CHR$ (4)
50 20 PRINT D$"OPEN TEXT2"
7A 30 PRINT D$"WRITE TEXT2"
EA 40 PRINT "COMPUTE!"
3A 50 PRINT "SUBSCRIPTION SERVICE"
D1 60 PRINT "P.O. BOX 10954"
E3 70 PRINT "DES MOINES"
E9 80 PRINT "IA 50340"
94 90 PRINT ""
91 100 PRINT "1-800-247-5470"
DA 110 PRINT D$"CLOSE"
```

Program 6. Joystick Modifications

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
FC 265 B0 = 4
CB 10150 X0 = PDL (0)
04 10160 Y0 = PDL (1)
78 10161 B1 = PEEK ( - 16287)
21 10162 IF B1 < 128 AND B0 = 3 THEN B0 = 4
21 10163 IF B1 < 128 AND B0 = 2 THEN B0 = 3
29 10164 IF B1 < 128 AND B0 = 1 THEN B0 = 3
25 10165 IF B1 > 127 AND B0 = 2 THEN B0 = 1
4D 10166 IF B1 > 127 AND B0 = 4 THEN B0 = 2
21 10170 IF PEEK ( - 16384) > 127 THEN 10440
A9 10180 Y0 = INT (Y0 / 10) + 1
8B 10190 X0 = INT (X0 / 6) + 1
32 20030 REM
```


Mousor

Escape Mode Cursor for the Apple IIc

J. Blake Lambert and Tim Victor

This short, fast utility makes it simple to use your Apple IIc mouse controller for editing in BASIC or the machine language monitor in escape mode. DOS 3.3 and ProDOS compatible.

Despite all the improvements Apple incorporated into the IIc, the screen editing features when using BASIC or the machine language monitor are not much better than those available on the IIe. Without an editing support package, it's difficult to copy and correct program lines. And there's no way to use the mouse controller to make editing easier.

In BASIC, usually you end up making corrections by just typing the incorrect line all over again. This wastes time and effort. The alternative is to use what is called *escape mode* editing.

"Mousor" makes using escape mode easy. By rolling the mouse over an area of the desk smaller than a 3 × 5-inch index card, you can cursor (mousor) anywhere on the screen.

How to Use Mousor

To start mousing with Mousor, type in and save the program below. It's a BASIC loader which POKes the Mousor machine language routine into memory. (Be sure to save the BASIC loader on disk before running it for the first time.) When you run Mousor, it automatically checks to see whether you're using DOS 3.3 or ProDOS and then adjusts itself accordingly. When the BASIC prompt reappears, you'll have a mouse-driven escape mode cursor. If you don't understand escape mode editing, see the instructions below.

The mouse is trained to work like this:

1. Click the mouse button to activate escape mode.
2. While holding the button down, roll the mouse across the desk to move the escape mode cursor.
3. Release the mouse button to exit escape mode.

Mousor locks out keypresses while it's in escape mode, so if you want to use the escape editing functions (like Esc-E to erase the end of a line), press the Esc key.

Getting a Line of BASIC

When you type a line of BASIC on the Apple IIc, a routine called GETLN puts the characters into a special area of memory called the input buffer. The first character on the line is stored at the start of the input buffer, and subsequent characters are added to the end of the buffer. This continues until you press the Return key to enter the line (with a few important exceptions). The computer clears the rest of the current screen line and stores the carriage return character into the input buffer to mark the end of the line.

When you make a mistake while entering the line, like typing the wrong character, it's easy to fix. For example, if you're entering a line such as `10 PRINT "HELLP"` and notice that you pressed P instead of O, you can press the left-arrow key (also called backspace) to back up and change the letter. Instead of storing the backspace in the buffer like other keypresses, GETLN treats it differently.

GETLN keeps track of the length of the input buffer by pointing to the end. When GETLN receives a backspace character, it lowers the value of the pointer by one. This removes the last character in the buffer, so all you need to do is continue typing.

If you don't notice your mistake until you've typed in several more characters, you can use the left- and right-arrow keys to make the correction. Press the backspace key until the cursor is on the letter you want to change. Type the correct letter, and then press the right-arrow key (also called *retype*) until the cursor returns to the end of the line.

Each time you press the retype key, the character currently under the cursor is added to the input buffer and the cursor is moved to the right. In effect, you have removed several characters from the buffer, changed the character you wanted to correct, and retrieved the rest of the characters one by one from the screen.

Now for the Tricky Stuff

Unfortunately, you can't always catch your typing errors before you press Return. Often, you don't even know there's a

problem in a line until you've run the program. Since the retype key allows you to pick up characters from the screen and add them to the input buffer, it would be handy if you could copy most of the bad line and type only the characters you want to change. This requires a way to move the cursor around the screen without affecting the input buffer.

Pressing the Esc key puts the IIc into escape mode. In this mode, the arrow keys move the cursor but don't change the input buffer. The IIc indicates escape mode by displaying a different cursor—a reverse plus sign. To leave escape mode, press Esc again.

Suppose you want to edit the following line in escape mode:

100 PRINT "THIS IS A TEDT"

If the line is not on the screen, you'll need to enter LIST 100. Press Esc to enter escape mode and move the cursor up to the 1. At this point, the input buffer is empty. Press Esc again and use the retype key to place the cursor on the *D*. This enters all but the last three characters into the input buffer. Now type the letter *S*, and press retype twice, followed by Return. If you like, list line 100 again to verify the correction.

To edit the same line with Mousor, you would LIST the line, click the mouse button and drag the cursor to the 1, and release the mouse button. After this, follow the same editing procedure.

Mouse Moves

You can also use escape mode to grab pieces of program lines. Mousor is especially adept at this, since movement is so easy and quick. Just keep in mind that when the mouse button is down no characters are added to the buffer.

To copy a line, first list it. Then enter the number for the new line you want to create, click the mouse button and drag the escape mode cursor to just beyond the original line number, and release the button. Copy the line by pressing the right-arrow key until you reach the last character and then press Return. Go to the new line number by clicking the mouse button and dragging the escape cursor to the right of the line number. Press Return and the original line material has been copied to this new line. (You can verify this by entering LIST *new line number*.)

Inserting is another useful technique. List the line first, then mousor (click and drag the escape mode cursor) to the beginning of the line number. Release the mouse button and use the right-arrow to move across the line until you reach the point where you want to insert characters. Press the mouse button and mousor to a blank line on the screen, then release the button and type the insert characters. Click and drag up to the listed line again, then release the button where you want the inserted material to go. Press Return, and the new additions are in place. (Again, type LIST line number to check that the changes have been made.)

Mousor

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
DE 10 IF PEEK (191 * 256) = 76 THEN GOSUB 40: GOSUB
    50: GOTO 30
AC 20 GOSUB 50: GOSUB 40
2E 30 FOR I = 11 TO 207: READ A: POKE I + 768,A: NEXT
    T : CALL 768: END
69 40 FOR I = 0 TO 10: READ A: POKE I + 768,A: NEXT
    : RETURN
13 50 FOR I = 0 TO 10: READ A: NEXT : RETURN
E7 60 DATA 216,169,67,141,50,190
4E 70 DATA 169,3,141,51,190
8E 80 DATA 169,67,133,56,169,3
7B 90 DATA 133,57,32,234,3
F7 100 DATA 120,162,196,160,64,32
77 110 DATA 28,196,169,0,141,120
7C 120 DATA 4,141,120,5,141,248
EA 130 DATA 5,169,8,141,248,4
0F 140 DATA 169,0,162,196,160,64
4C 150 DATA 32,176,196,169,1,162
02 160 DATA 196,32,176,196,162,196
EF 170 DATA 32,132,196,169,1,162
52 180 DATA 196,160,64,32,61,196
D6 190 DATA 88,96,145,40,32,76
66 200 DATA 204,44,99,192,16,8
51 210 DATA 32,112,204,16,246,76
2B 220 DATA 37,253,218,90,72,169
74 230 DATA 4,141,124,4,141,252
FC 240 DATA 4,32,187,3,32,195
FB 250 DATA 3,44,99,192,16,9
6A 260 DATA 104,32,179,195,122,250
82 270 DATA 76,69,3,173,124,4
B4 280 DATA 240,7,201,8,144,25
5C 290 DATA 162,156,44,162,136,104
```

E5 300 DATA 32,179,195,138,32,88
3A 310 DATA 205,32,195,3,72,169
A3 320 DATA 4,141,124,4,32,187
40 330 DATA 3,173,252,4,240,7
DA 340 DATA 201,8,144,25,162,138
5F 350 DATA 44,162,159,104,32,179
82 360 DATA 195,138,32,88,205,32
08 370 DATA 195,3,72,169,4,141
C7 380 DATA 252,4,32,187,3,76
1F 390 DATA 102,3,162,196,160,64
C9 400 DATA 32,107,196,96,32,29
60 410 DATA 204,72,41,128,73,171
15 420 DATA 32,179,195,104,96

6

Programming and Utilities



Keynote

Patrick Parrish

Many personal computers offer the user some sort of aural feedback with each keystroke. "Keynote" makes this capability available for the Apple II series computer using either DOS 3.3 or ProDOS.

Most of us can't touch-type. We may not be of the "hunt and peck" school of typists, but we don't blitz-type at 60 words per minute either. We're somewhere in the middle.

So auditory feedback while we type is something we appreciate. Some computers provide this, a blip or a beep. It makes it easier to tell when a key's been pressed. Until now, the Apple II computers didn't have this feature.

With "Keynote," you'll hear a tone with each keypress. It doesn't matter whether you're in immediate mode or within a program. Furthermore, this tone can be altered to suit your ears or even turned off.

Short and Simple

Keynote is a short 52-byte machine language routine loaded into the Apple by a BASIC loader. Carefully type in the program listing below and save a copy to disk before you run it. When you run the BASIC loader, line 100 POKES the machine language code into memory, starting at location 924 (that's \$39C in hexadecimal notation). This area is safe from BASIC on the Apple, so Keynote shouldn't interfere with or be overwritten by most programs. Next, line 110 saves a copy of Keynote as a binary file (entitled KEYNOTE) to disk, then ends the program. (Because the BASIC program uses the name KEYNOTE for the machine language file it writes to disk, you must use some other name when you save the BASIC program. If you save the BASIC program with the name KEYNOTE, you'll get a FILE TYPE MISMATCH error when you run it.)

Keynote is now in memory. To activate it, type CALL 924 and press Return. Now press any key, and you should immediately hear a click from the speaker.

You may find this tone annoying, especially after typing for long stretches. Fortunately, Keynote lets you use other

notes. Finding a tone to suit your own ear may take some trial and error testing, however. To try other tones, simply POKE values for pitch and duration (both must be in the range 0–255) into locations 974 and 975, respectively, and press a few keys. By default, the pitch has a value of 100 and duration of 10 (see line 180 in the BASIC loader). Once you've settled on a suitable tone, change the values in line 180, and rerun the BASIC loader. Keynote will thus be resaved with the values you choose. After you have a tone you like, you won't need the BASIC program again—you can just use the binary file it creates. This is explained in detail in the following section.

Note: If you choose a long duration, you'll notice that characters are displayed more slowly than they're entered when you're typing quickly. The tone may also seem to be continuous. It's best to use a relatively low value for the duration.

If you tire of Keynote's incessant clicks altogether, just hit Ctrl-Reset to deactivate it. To run it again, simply CALL 924.

On Your Own

Since Keynote is saved to disk as a binary file, it can easily be loaded and run by other programs, or run from immediate mode. For example, DOS 3.3 automatically boots up any program named HELLO on the disk. To have KEYNOTE load and run when you boot up your disk, simply include this statement in your HELLO program:

```
PRINT CHR$(4)"BRUN KEYNOTE"
```

Likewise, to run Keynote from immediate mode with DOS 3.3, just type

```
BRUN KEYNOTE
```

If you're using ProDOS, Keynote requires that you first enter BLOAD KEYNOTE, then follow it with a CALL 924.

This is necessary since the pointers directing the operating system to Keynote are reset by ProDOS after a BRUN or BLOAD. Thus, if you want a BASIC program to automatically run Keynote from ProDOS, just include this statement:

```
PRINT CHR$(4)"BLOAD KEYNOTE":CALL 924
```

Of course, you can use Keynote from immediate mode in ProDOS, too. Just type

```
BLOAD KEYNOTE: CALL 924
```

How It Works

Keynote works much the same in both DOS 3.3 and ProDOS. In both operating systems, a *wedge* is used. The input vectors which normally point to the keyboard input subroutine (KEYIN) at location 64795 (\$FD1B) are changed to point to the starting location of Keynote. Once this is done, Keynote jumps to KEYIN. KEYIN waits for a key to be pressed, and when one is, the routine returns control to Keynote. This produces a tone and, in turn, returns to BASIC.

Before all this can happen, though, Keynote must go through a short initialization routine to determine which operating system is being used. This is done by looking at location 48896 (\$BF00), the starting location for ProDOS's global page. When ProDOS boots, the value in location 48896 is always 76 (representing the JMP instruction). If 76 is indeed the value at this location, the vectors pointing to KEYIN (CHIN1 at 48690-48691 [\$BE32-\$BE33]) are loaded with the Keynote's starting address (949 [\$3B5]) in low-byte/high-byte format, and control is returned to BASIC.

If the value at location 48896 is anything other than 76, Keynote assumes it's in DOS 3.3. In this case, the input vectors (KSW, for KeySWitch) at 56-57 (\$38-\$39), which normally point to KEYIN, are loaded with Keynote's starting address. A jump to a routine at 1002 (\$3EA) updates the input pointers with these new values, reconnects DOS, and returns to BASIC. With either operating system, Keynote is called, and a tone is generated with each keypress.

Keynote BASIC Loader

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```

10 100 FOR I = 924 TO 975: READ A: POKE I,A: NEXT
11 110 PRINT CHR$ (4)"BSAVE KEYNOTE,A$39C,L$34": END
12 120 DATA 162,181,160,3,173,0,191,201
13 130 DATA 76,208,7,142,50,190,140,51
14 140 DATA 190,96,134,56,132,57,76,234
15 150 DATA 3,32,27,253,32,74,255,172
16 160 DATA 207,3,174,206,3,173,48,192
17 170 DATA 202,208,253,136,208,244,32,63
18 180 DATA 255,96,100,10: REM LAST 2 VALUES ARE PIT
    CH, DURATION

```


Easy Apple Screen Editing

Roland Brown

Enhanced version by Tim Victor

Here's a way to make BASIC programming easier and more fun: an advanced screen editor that makes up for the Apple's lack of full-screen editing. This outstanding programming utility works on any Apple II series computer using either DOS 3.3 or ProDOS, in 80-column as well as 40-column mode.

Although Applesoft BASIC is a powerful language, its screen editor leaves much to be desired. Some Apple II owners invest in a ROM editor, others write their programs with a word processor, and the rest just suffer with the frustrating ESCape mode editing. But ROM editors cost money, word processors don't let you flip back and forth between the text editor and BASIC to test changes, and suffering isn't *always* good for the soul. So here's a better solution: "BASIC Line Editor," a powerful utility that lets you easily modify BASIC program lines.

Because the BASIC Line Editor program is written entirely in machine language, it must be entered with the "Apple MLX" machine language entry program found in Appendix C. Be sure you read and understand the instructions for using MLX before you begin entering the data. When you run MLX, you'll be asked for a starting address and an ending address for the data you will be entering. For BASIC Line Editor, the proper values are as follows:

Starting address: 2000

Ending address: 23C6

When you have entered all the data, be sure to use the Save option to save at least one copy before you leave MLX.

Once you've entered all the data and saved a copy, you're ready to use the BASIC Line Editor. Start it by typing *BRUN filename* and pressing Return (substitute the filename you used when you saved the BASIC Line Editor data with MLX). The program loads at memory address \$2000 (that's 8192 in decimal notation), then checks to see which operating system is present before moving itself to a safe location. This process

can destroy part of a long BASIC program. If you have a long BASIC program in memory, you should save it *before* you activate the BASIC Line Editor.

Now you're ready to put the Editor to work. To edit a BASIC program line, type the ampersand (&) followed by the desired line number. For instance, enter &100 to edit line 100. The BASIC Line Editor displays the line on the screen in a format somewhat different from Applesoft's. The line is continuous rather than centered on the screen, there are no extra spaces in the line except between quotation marks, and all control characters are displayed in reverse video.

(You can access only one program which uses the ampersand at a time. Thus, you can't use the BASIC Line Editor at the same time as the "Keyboard Customizer" or "Apple Hi-Res Screen Dump," also in this book.)

Editing Commands

The BASIC Line Editor provides 13 new editing functions. Most are accessed by pressing the Ctrl (Control) key along with a letter key. Here's a quick reference table followed by a detailed description of each command:

Key	Function
Ctrl-B	Block back
Ctrl-C	Convert hex to decimal
Ctrl-D	Delete right
Ctrl-F	Block forward
Ctrl-H	Cursor left
Ctrl-I	Insert
Ctrl-M	Return
Ctrl-S	Search
Ctrl-T	Truncate
Ctrl-U	Cursor right
Ctrl-V	Verbatim
Delete	Delete left
Esc	Return to BASIC

Ctrl-B (block back) moves the cursor back to the previous colon, or if there is no previous colon, to the beginning of the line.

Ctrl-C (convert hex) converts hexadecimal numbers to decimal. This command moves the cursor above the line being edited, prints a \$ prompt on the screen, and waits for you to enter a number. This value is converted to decimal and printed. Then the cursor returns to its original position on the line.

Ctrl-D (delete right) deletes the character under the cursor. The cursor stays where it is and everything to the right moves back one space.

Ctrl-F (block forward) moves the cursor forward to the next colon, or if there is no colon, to the end of the line.

Ctrl-H (cursor left) moves the cursor back one space.

Ctrl-I (insert) puts the BASIC Line Editor in insert mode. Any characters you type are inserted in the line until you use another Editor command.

Ctrl-M (return) is the same as pressing Return. No matter where the cursor is located on the line, pressing Ctrl-M enters the line into the program.

Ctrl-S (search) searches for the next character entered. If the search fails, in other words, if there is no character specified from the cursor to the end of the line (or if there's no such character in the line at all), the cursor moves to the end of the line.

Ctrl-T (truncate) truncates the line at the cursor position (deletes everything after the cursor). The cursor ends up one space beyond the new end of the line.

Ctrl-U (cursor right) moves the cursor forward one space.

Ctrl-V (verbatim) lets you enter control characters verbatim. If the keypress immediately after Ctrl-V is a Control key combination, it's interpreted as a control character rather than as a BASIC Line Editor command. Ctrl-V is useful for adding Return (Ctrl-M) or backspace (Ctrl-H) characters to a line for improved printing control. If the keypress immediately following Ctrl-V is not a Control key combination, Ctrl-V has no effect. Remember that the BASIC Line Editor shows control characters in reverse video.

Delete (delete left) deletes the character to the left of the cursor and moves the cursor back one space. (The Delete key is found only on the Iie and Iic.)

Esc (return to BASIC) puts you back in BASIC. If you make a mistake when editing a line with the BASIC Line Editor, press Esc to exit back to BASIC without losing the line.

Program Notes

Activating the Editor resets the stack to the same level as does BASIC, sets up the ampersand vector (\$3F5), moves the DOS buffers downward to protect DOS, and restarts BASIC. The Editor uses existing BASIC routines to read the input line and

find the desired line in memory. If you try to edit a line that doesn't exist, the Editor simply returns to BASIC. If the line is found, its contents are read and listed on the screen. Text characters are listed just as they are stored. When the Editor finds a token (an encoded BASIC keyword), it locates the word in the BASIC keyword table and lists it on the screen.

Once the Editor lists the line, it enters editing mode. This part of the program gets a command from the keyboard, processes it, and updates the screen. Space doesn't permit a detailed explanation of how each Editor command works. If you're familiar with Apple machine language programming, you may find it interesting to trace through the various routines on your own.

BASIC Line Editor

For mistake-proof program entry, use the "Apple MLX" (Appendix C) to type in this program.

START ADDRESS: 2000
END ADDRESS: 23C6

```

2000: AD 00 BF C9 4C D0 0D A9 15
2008: 03 20 F5 BE 18 A5 74 69 26
2010: 04 4C 1B 20 38 A5 74 E9 F5
2018: 03 85 74 85 CF 8D AF 20 56
2020: A5 73 85 CE 8D AE 20 A9 BE
2028: B1 85 EB A9 20 85 EC A0 4C
2030: 00 B1 EB 91 CE E6 CE D0 F3
2038: 02 E6 CF E6 EB D0 02 E6 29
2040: EC A5 EB C9 46 D0 EA A5 6B
2048: EC C9 23 D0 E4 B1 EB E6 8F
2050: EB D0 02 E6 EC 8D B0 20 88
2058: 11 EB F0 29 AD B0 20 18 55
2060: 6D AE 20 85 CE B1 EB E6 5B
2068: EB D0 02 E6 EC 6D AF 20 1E
2070: 85 CF 18 B1 CE 6D AE 20 2F
2078: 91 CE C8 B1 CE 6D AF 20 15
2080: 91 CE 88 F0 C8 AD AE 20 D7
2088: 8D F6 03 AD AF 20 8D F7 99
2090: 03 A9 4C 8D F5 03 A0 0B 27
2098: B9 A2 20 20 F0 FD 88 10 05
20A0: F7 60 8D D9 C4 C1 C5 D2 CF
20A8: A0 B2 C5 CC C2 8D 38 20 48
20B0: 20 20 0C DA 20 1A D6 B0 FF
20B8: 01 60 68 68 20 9C FC A0 33
20C0: 02 B1 9B C8 AA B1 9B 20 E1
20C8: 24 ED A0 06 8C 7B 05 84 EB
20D0: CE A5 25 8D 97 02 A5 9B 0B

```



```

20D8: 85 EB A5 9C 85 EC A0 04 7A
20E0: B1 EB C8 C9 00 F0 2C 10 D6
20E8: 24 A2 D0 8E 44 00 8E 45 6B
20F0: 00 29 7F AA AD FF FF 30 B3
20F8: 11 E0 00 D0 03 20 48 01 31
2100: EE 44 00 D0 EF EE 45 00 9D
2108: D0 EA CA 10 F3 20 48 01 79
2110: 38 B0 CD A0 06 A9 C0 8D 44
2118: 98 02 84 CF 20 22 01 20 60
2120: 0C FD C9 FF D0 02 A9 80 83
2128: C9 A0 90 51 2C 98 02 30 96
2130: 0F 70 41 8D 46 01 20 23 B0
2138: 02 A9 C0 8D 98 02 30 DA DE
2140: 70 22 48 A4 CF 84 E3 A4 93
2148: CE 8C 95 02 C8 20 EC 01 89
2150: 84 CF 20 60 01 CE 95 02 43
2158: C6 CF A4 E3 C4 CF D0 F2 BE
2160: 20 22 01 68 20 6E 01 A4 43
2168: CF C4 CE C8 90 03 20 EC E7
2170: 01 4C 69 00 A4 CF A9 C0 EB
2178: 8D 98 02 30 9D 2C 98 02 BB
2180: 30 0D 50 F0 A2 C0 8E 98 05
2188: 02 49 C0 C9 40 D0 D5 A2 66
2190: C0 8E 98 02 C9 8D F0 0C 7C
2198: C9 98 F0 2E A4 CF 20 FD 4A
21A0: 01 4C 69 00 A0 00 84 CF 81
21A8: 20 22 01 20 9B 01 49 80 99
21B0: 10 02 29 3F A4 CF 99 00 2C
21B8: 02 C8 C4 CE D0 E8 A9 00 31
21C0: 99 00 02 A0 01 A2 FF 4C F8
21C8: 44 D4 A4 CE 20 22 01 A0 10
21D0: 00 F0 EB 48 AD 97 02 85 A6
21D8: 25 98 C5 21 90 06 E5 21 28
21E0: E6 25 B0 F6 85 24 8D 7B B8
21E8: 05 20 22 FC 68 60 84 CF 67
21F0: 20 22 01 20 9B 01 C9 46 A8
21F8: 60 8C 96 02 09 80 C9 A0 FF
2200: B0 02 49 C0 20 6E 01 A4 B3
2208: CE C8 20 EC 01 AC 96 02 A2
2210: 60 AC 95 02 20 22 01 20 2E
2218: 9B 01 A4 CF 20 22 01 8D 15
2220: 99 02 A5 25 48 AD 7B 05 AD
2228: 85 24 48 AD 99 02 20 F0 22
2230: FD 68 CD 7B 05 D0 07 C5 3E
2238: 24 A5 24 8D 7B 05 68 90 A6
2240: 07 C5 25 D0 03 CE 97 02 AF
2248: AD 99 02 60 AD 7B 05 AC 22
2250: B3 FB C0 06 D0 16 2C 1F 3C
2258: C0 10 11 8D 01 C0 48 38 CF
2260: 65 20 4A 90 03 2C 55 C0 E5

```

```

2268: 68 69 00 4A A8 B1 28 2C 68
2270: 54 C0 60 C0 00 F0 25 20 55
2278: F7 01 84 CF 84 E3 20 0F 89
2280: 02 8C 95 02 C4 CE F0 0D 0C
2288: 20 60 01 EE 95 02 E6 CF 56
2290: AC 95 02 D0 EF A4 CF 20 AF
2298: EC 01 A4 E3 60 84 CE 20 39
22A0: 22 01 20 9C FC A4 CE 60 7C
22A8: C0 00 F0 01 88 60 A2 0B 91
22B0: CA 30 FA DD 7F 02 D0 F8 42
22B8: BD 8A 02 8D 0E 02 B0 FF 71
22C0: C4 CE F0 01 C8 60 A9 80 E4
22C8: 2C A9 00 2C A9 40 8D 98 52
22D0: 02 60 A9 BA 8D 46 01 A4 3B
22D8: CF C4 CE F0 06 C8 20 3D EF
22E0: 01 D0 F4 A4 CF 60 A9 BA D0
22E8: 8D 46 01 A4 CF F0 06 88 C6
22F0: 20 3D 01 D0 F6 A4 CF 60 0C
22F8: AC 97 02 88 84 25 20 22 5D
2300: FC A9 00 8D 7B 05 20 9C D4
2308: FC A2 00 A9 A4 20 6E 01 93
2310: 20 0C FD 9D 00 02 E8 C9 A6
2318: 8D D0 F2 20 C7 FF 20 A7 DF
2320: FF A9 BD 20 F0 FD A5 3F 94
2328: A6 3E 20 24 ED A4 CF 60 99
2330: 80 84 88 95 94 89 93 96 CA
2338: 86 82 83 B3 BC E8 00 DD 75
2340: 06 09 0C 12 26 38 23 00 C6
2348: 3B 00 3E 00 4D 00 50 00 FE
2350: 55 00 5D 00 67 00 6C 00 01
2358: 7C 00 83 00 86 00 8B 00 98
2360: 99 00 9D 00 A2 00 A5 00 87
2368: B0 00 B4 00 BE 00 C1 00 17
2370: C8 00 CD 00 D6 00 E1 00 4F
2378: EE 00 F1 00 F8 00 FB 00 34
2380: 1C 01 24 01 40 01 43 01 37
2388: 49 01 54 01 5A 01 5D 01 E0
2390: 61 01 64 01 67 01 6C 01 7D
2398: 6F 01 7B 01 95 01 99 01 39
23A0: C7 01 CE 01 D1 01 D8 01 3A
23A8: DB 01 E0 01 E7 01 EF 01 6D
23B0: 03 02 08 02 0B 02 1E 02 B8
23B8: 24 02 2E 02 38 02 40 02 C3
23C0: 48 02 5D 02 00 00 24 49 09

```

Apple Disk Booster

D. W. Hoover

This unusual BASIC program increases the amount of storage space on Apple disks in DOS 3.3. It runs on any Apple II series computer with a disk drive. For DOS 3.3 only.

If you use a disk drive, you know that disk space is a precious commodity. One way to increase disk storage is to buy special hardware. But that's a costly proposition. "Apple Disk Booster" offers a simple, inexpensive alternative. It lets you format new disks with up to five extra tracks, creating more than 21,000 bytes of extra storage space per disk.

Type in Apple Disk Booster and save a copy before you run it. First, the program prompts you to insert a blank disk in drive 1, then it initializes the disk. Because different drives allow a different number of extra tracks, Apple Disk Booster formats only as many extra tracks as your drive can reliably use. The program automatically reads and verifies each extra track. If a track cannot be used, restart the initialization using the next lower track value. When it finishes the initialization, the program displays the number of tracks formatted on that disk.

Since Disk Booster is now the HELLO program on the disk, delete it by typing `DELETE DISK BOOSTER,D1` and pressing Return. (This prevents you from accidentally running it again.) The disk is now ready for normal use.

As noted above, different drives may not be able to use the same number of extra tracks. If you want to use your modified disk on a different drive, it's a good idea to determine beforehand whether the drive can access the extra tracks. To do this, simply run Apple Disk Booster on the second drive and note the number of tracks displayed when the program ends. Once you know the number of tracks that both drives can access, substitute that number for 40 in line 60 of the program, and run it again as needed.

If you later need to transfer files to a normal disk, use the DOS FILEM utility on the Apple DOS 3.3 *System Master* disk.

Extra Tracks

Squeezing extra tracks onto an Apple disk is surprisingly easy to do. This program modifies values used by the DOS routines

that initialize the disk and create its Volume Table of Contents (VTOC). Apple disks are normally formatted with 35 tracks. The first POKE in line 130 forces DOS to format more tracks by substituting a larger number-of-tracks value.

The remaining POKEs in that line adjust the VTOC and bitmap accordingly. The bitmap is a portion of the VTOC that shows where free sectors are located on the disk. Each track has four bytes in the bitmap (two bytes are never used), and each bit represents a corresponding sector in the track. If a bit is off (set to 0), the sector is already allocated. If a bit is on (1), the sector is free. Here is the general format of the VTOC and bitmap:

Byte	Description
00	Not used
01	Track of first catalog sector
02	Sector of first catalog sector
03	DOS release number (3.3, etc.)
04-05	Not used
06	Volume number
07-26	Not used
27	Max number of track/sector pairs
28-2F	Not used
30	Last allocated track
31	Direction of allocation
32-33	Not used
34	Number of tracks per disk
35	Number of sectors per track
36-37	Number of bytes per sector
38-3B	Bitmap of track 0
3C-3F	Bitmap of track 1
40-43	Bitmap of track 2
...	...
BC-BF	Bitmap of track 33
C0-C3	Bitmap of track 34
C4-FF	Bitmaps of additional tracks (if desired)

To insure that the VTOC and bitmap accommodate the extra tracks, the last two POKE statements in line 130 set new values for the number of tracks on the disk and the size of the bitmap. If 40 tracks are formatted, the bitmap is 160 (40×4) bytes in size. Of course, it's important to be sure the disk drive can use the extra tracks reliably. Lines 440-510 of Disk Booster contain data for a machine language routine that checks the new tracks. It reads a random sector from each ex-

tra track and checks for read-back errors. If an error occurs, it's assumed that the track cannot be accessed and the disk is reinitialized without that track.

Apple Disk Booster

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
05 100 TRACKS = 40: REM # OF TRACKS VALUE
F5 110 BITMAPSZ = TRACKS * 4: REM BIT MAP SIZE
53 120 FOR ML = 768 TO 829: REM LOC OF ML ROUTINE
58 130 READ BYTE: POKE ML,BYTE:CHKSUM = CHKSUM + BYT
    E
01 140 NEXT
51 150 IF CHKSUM < > 9911 THEN HOME : PRINT "ERROR I
    N DATA STATEMENTS": GOTO 260
88 160 POKE 216,0: ONERR GOTO 270: REM RESET ONERR F
    LAG
A1 170 POKE 48894,TRACKS: POKE 46063,TRACKS: POKE 44
    725,BITMAPSZ
E8 180 HOME : PRINT "FORMATTING ";TRACKS;" TRACKS...
    "
F7 190 PRINT CHR$ (4);"INIT DISK BOOSTER,D1"
C9 200 CALL 768: REM CALL VERIFY TRACKS ML ROUTINE
    AT $0300
58 210 PRINT "FORMATTING COMPLETE... ";TRACKS;" TRAC
    KS WRITTEN"
E5 220 PRINT
8A 230 PRINT "DON'T FORGET TO LABEL"
88 240 PRINT "YOUR NEW DISK WITH"
9F 250 PRINT "THE NUMBER OF TRACKS INITIALIZED!"
96 260 END
70 270 A = PEEK (222): REM ERROR CODE
76 280 ERRL = PEEK (218) + PEEK (219) * 256: REM ERR
    OR LINE
F5 290 IF A < > 8 THEN GOTO 350
84 300 HOME : PRINT "ERROR DETECTED..."
78 310 PRINT "RESETTING NEW TRACK VALUE"
EA 320 PRINT "PRESS <RETURN> TO CONTINUE"
08 330 INPUT "":IN$: IF IN$ < > "" THEN GOTO 330
31 340 TRACKS = TRACKS - 1:BITMAPSZ = TRACKS * 4: GO
    TO 160
E3 350 PRINT "ERROR NUMBER ";A;" DETECTED IN LINE ";
    ERRL
83 360 PRINT "CHECK DOS PROGRAMMERS MANUAL"
65 370 PRINT "FOR ERROR TYPE"
95 380 PRINT "AND CORRECT IT ACCORDINGLY"
25 390 GOTO 260
E2 400 REM **ERROR CODES FOR APPLE DOS 3.3**
87 410 REM
```

```
79 420 REM ERROR #4 = WRITE PROTECTED DISK (REMOVE  
    PROTECT TAB)  
AE 430 REM ERROR #11= SYNTAX ERROR (CORRECT TYPOS)  
AF 440 DATA 169,0,141,235,183,141,240,183  
AD 450 DATA 169,1,141,244,183,169,16,141  
0C 460 DATA 241,183,133,209,173,254,190  
5E 470 DATA 141,236,183,56,233,35,144,22,133  
94 480 DATA 210,169,183,160,232,32,181  
1B 490 DATA 183,176,12,206,236,183,198  
3F 500 DATA 210,208,240,198,209,208,224  
8A 510 DATA 96,169,8,141,92,170,76,213,166
```


Apple Keyboard Customizer

Robert Buehler

This interesting program lets you reconfigure your Apple keyboard, even save the changes on disk for future use. It works on any Apple II series computer with DOS 3.3 only.

Are you frustrated with the Apple keyboard? Are you tired of the way Apple arranged the keys? Do you yearn for a numeric keypad? If so, "Keyboard Customizer" may be for you. It lets you rearrange your keyboard anyway you want.

You can convert part of the regular keyboard into a numeric keypad—and even make a hexadecimal pad if you desire. This pad can be laid out using the keys of your choice.

Do you keep missing the Return key and wish it were larger? No problem. Define three keys as Returns.

Besides letting you add such things as a numeric pad, Keyboard Customizer gives you the opportunity to eliminate pet annoyances. For instance, the colon (:) is commonly used when typing Applesoft BASIC programs. As the regular keyboard is set up, the semicolon and colon share the same key. To enter a colon, you must press Shift. With Customizer, the positions of these two characters could be reversed.

The question mark is another familiar character for Applesoft programmers as an abbreviation for PRINT. Using Keyboard Customizer, you could reposition the question mark to the semicolon key, making it more accessible. All of these—and any other modifications that fit your fancy—are at your fingertips with Keyboard Customizer.

Typing the Program

To prepare Keyboard Customizer, you must type it in with "Apple MLX," the machine language entry program found in Appendix C. MLX catches most typing mistakes as they happen and helps insure that you'll finish with an error-free copy. Read the MLX instructions carefully before you begin.

Note: Before loading MLX, type this line:

HIMEM: 31744

Press Return. Now load MLX and respond to the starting and ending address prompts:

Starting address: 8000

Ending address: 81A1

When you finish typing the listing, MLX prompts you to save a copy on disk.

Four Customizer Commands

To run Keyboard Customizer, type BRUN KEYBOARD (or whatever filename you specified when you saved the program with MLX). The] prompt should appear as usual.

Keyboard Customizer has four commands, which must be preceded by the ampersand (&). (*You can access only one program which uses the ampersand at a time. Thus, you can't use Keyboard Customizer at the same time as the "BASIC Line Editor" or "Apple Hi-Res Screen Dump," also in this book.*)

Command	Function
&0	Restores the keyboard to its original configuration (as does pressing the Reset key or a reboot)
&1	Activates the customized keyboard
&2	Enters the keyboard editor
&3	Prints a list of key values in the format <i>original key = customized key value</i>

All these commands are pretty much self-explanatory except &2, which calls up the keyboard editor. This is the tool for altering the key values. The first thing you notice after typing &2 is the message *FIRST KEY:*. The program is asking you to begin defining the range of keys you want to customize.

The editor looks at keys sequentially by their ASCII codes. ASCII (American Standard Code for Information Interchange) is a system which assigns numbers to standard characters which appear on computer keyboards. The ASCII code for an uppercase *A*, for example, is 65; *B* is 66; *C* is 67; and so on. All letters, numbers, punctuation marks, and other symbols have an ASCII code; a table of these codes can be found in the *Apple II User's Guide* or in just about any other computer manual. You can also determine the ASCII value of a character in BASIC by typing `PRINT ASC("A")`, substituting the appropriate character for *A*.

To specify a range of keys, first find the ASCII value of the *lowest-numbered* character you want to customize. Make

sure this is the key you press at the FIRST KEY: prompt. Then find the ASCII value of the *highest-numbered* character you want to customize. *This is the key you should press at the LAST KEY: prompt.* (Therefore, the ASCII value of the key you press at the FIRST KEY: prompt should always be equal to or less than the ASCII value of the key you press at the LAST KEY: prompt.) Any character can begin or end the range, including Escape or Control characters. You'll notice that Control characters along with Escape are displayed in reverse video for easier identification.

After entering the starting and ending keys of your editing range, you'll see the message ENTER THE NEW REPLACEMENT VALUE FOR EACH KEY. The program displays the first character in the range you specified, followed by a colon. Next, enter the new replacement character. Do not press Return—Keyboard Customizer automatically enters a carriage return and then prompts you with the next key to be edited.

When you've assigned new values to all the keys in the range, the program returns to BASIC. Try typing one of the keys you have altered. It should return the reassigned character. Enter a command using that key, or write a program using the key. Even in PRINT and INPUT statements, the key yields its new character value.

How It Works

It seems as though Keyboard Customizer brings about some drastic changes. Actually, it doesn't. To understand how the program works, let's review how the Apple handles keyboard input.

Every time a key is pressed, Applesoft BASIC looks at memory locations \$38-\$39, its input hook. These locations normally contain the address of KEYIN (\$FD1B), a routine in read only memory (ROM) that gets the keypress from the keyboard. However, the input hook can be made to point to an alternative input routine. This is the case with Keyboard Customizer. Control passes not to the KEYIN routine in ROM, but rather to a routine within Customizer. This routine calls KEYIN to get the character code for the keypress, but checks to see if the code belongs to a character that was altered. If so, Customizer replaces it with the customized value.

The part of Customizer which replaces the old key values is actually very short (only five bytes). A much larger part of

the program is the buffer it uses to store the modified values. Along with the editor, the buffer comprises the majority of the program. The buffer is so large because it stores the values for all the keys sequentially, even if they equal the original values. As a result, the buffer size is constant—half a page of memory (128 bytes). It may seem a waste of memory to store the values of keys which haven't been changed. But if only modified keys were stored in the buffer, the routine that replaces the character values would be much longer and more complicated.

Save the Keyboard

This brings up another important point. Keyboard Customizer's improvements are temporary, since the input hook at \$38-\$39 is initialized during a reset or reboot. But there's a way to save the keyboard changes you've made. First, enter the Apple's built-in machine language monitor by typing `CALL -151`. Then type this line and press Return:

8016: EA EA EA

This stops Keyboard Customizer from clearing the buffer by overwriting three machine language instructions with NOPs (No Operation, similar to REM in BASIC). Second, you'll need to save the buffer that holds all the modifications along with the original program. Enter this command:

BSAVE KEYBOARD1,A,\$8000,L\$23C

To run this new version, simply type `BRUN KEYBOARD1`. You could also include the command `BRUN KEYBOARD1` in the HELLO program so that the customized keyboard automatically loads every time you boot the system.

Keyboard Customizer Routines and Important Locations

AMPERV	\$3F5	Holds JMP instruction for & commands
CH	\$24	Cursor horizontal displacement
COUT	\$FDF0	Prints byte in accumulator onscreen
CRDO	\$DAFB	Prints a carriage return
CV	\$25	Cursor vertical position
DOSHOOK	\$3EA	Connects I/O hooks to DOS
GETBYT	\$E6F8	Evaluates formula at TXTPTR
KEYIN	\$FD1B	Gets next key input from keyboard
KSWL	\$38-\$39	DOS input hook
RDKEY	\$FD0C	Call KEYIN via KSWL

Apple Keyboard Customizer

For mistake-proof program entry, use "Apple MLX" (Appendix C) to type in this program.

START ADDRESS: 8000

END ADDRESS: 81A1

```

8000: A2 4C A0 2F A9 80 8E F5 CE
8008: 03 8C F6 03 8D F7 03 A0 AF
8010: 4C A9 81 20 3B 81 20 2B 1F
8018: 81 20 6F FB A0 27 A9 80 05
8020: 84 38 85 39 4C EA 03 20 E9
8028: 1B FD A8 B9 21 81 60 20 D6
8030: F8 E6 E0 00 D0 0A A0 57 CA
8038: A9 81 20 3B 81 4C 19 81 17
8040: E0 02 D0 69 20 19 81 A0 EB
8048: 86 A9 81 20 3B 81 20 0C 55
8050: FD 8D 9E 81 20 23 81 20 50
8058: F0 FD A0 91 A9 81 20 3B 4D
8060: 81 20 0C FD 8D A0 81 20 9D
8068: 23 81 20 F0 FD A0 5C A9 43
8070: 81 20 3B 81 20 19 81 AD CF
8078: 9E 81 20 23 81 20 F0 FD CB
8080: A9 BA 20 F0 FD A9 A0 20 10
8088: F0 FD 20 0C FD 8D 9F 81 2D
8090: 20 23 81 20 F0 FD AC 9E 14
8098: 81 AD 9F 81 99 21 81 20 46
80A0: FB DA CC A0 81 F0 60 EE 79
80A8: 9E 81 4C 77 80 E0 03 D0 B8
80B0: 4B 20 58 FC 20 FB DA AD 8E
80B8: 9F 81 85 24 AD 9E 81 20 E7
80C0: 23 81 20 F0 FD A9 BD 20 F8
80C8: F0 FD AE 9E 81 BD 21 81 48
80D0: 20 23 81 20 F0 FD EE 9E D8
80D8: 81 20 FB DA AD 9E 81 C9 84
80E0: DF F0 24 A5 25 C9 14 D0 36
80E8: CE A9 00 85 25 A9 08 18 0C
80F0: 6D 9F 81 8D 9F 81 20 FB D8
80F8: DA 4C B7 80 A0 54 A9 81 A4
8100: 20 3B 81 4C 1C 80 60 A9 24
8108: 00 8D 9F 81 A9 80 8D 9E 83
8110: 81 A9 DE 8D A0 81 4C 1C B2
8118: 80 20 89 FE 20 93 FE 20 F1
8120: EA 03 60 C9 A0 10 03 38 85
8128: E9 80 60 A9 80 A2 00 9D 13
8130: A1 81 E8 A8 C8 98 C9 FF 48
8138: D0 F5 60 84 06 85 07 A0 6A
8140: 00 B1 06 F0 06 20 F0 FD 10
8148: C8 D0 F6 60 8D D2 C5 C1 CD
8150: C4 D9 8D 00 CF CE 00 CF 67
8158: C6 C6 87 00 8D 8D C5 CE 5E

```

```

8160: D4 C5 D2 A0 D4 C8 C5 A0 99
8168: CE C5 D7 A0 D2 C5 D0 CC 65
8170: C1 C3 C5 CD C5 CE D4 8D 7B
8178: C6 CF D2 A0 C5 C1 C3 C8 BC
8180: A0 CB C5 D9 8D 00 C6 C9 E0
8188: D2 D3 D4 A0 CB C5 D9 BA 72
8190: 00 A0 A0 A0 CC C1 D3 D4 C3
8198: A0 CB C5 D9 BA 00 80 00 0C
81A0: DE 00 20 4D 41 52 47 4F 1D

```


Instant Apple Help Screens

Kent Brewster

Using this short utility, you can design and save your own custom help screens to disk, then quickly call them into BASIC programs. For all Apple II series computers with DOS 3.3 or ProDOS.

As professional software designers have discovered, help screens are very popular features in all kinds of programs. Users don't have to fumble around with "handy" reference cards—they can just hit Esc or some other key and call up a screenful of instructions.

"Help Screen Editor," listed below, is a utility program that lets you create help screens of your own which are then saved to disk as binary files. Once saved, the file can be summoned back to the screen with a simple BLOAD command. If you BLOAD the file to an area of memory known as text page 2 with this statement:

BLOAD filename,A\$800

your help screen can be viewed and swapped with text page 1 (the normal screen) with this statement:

POKE -16299,0

The following statement will switch back to the normal screen display (text page 1):

POKE -16300,0

By placing these lines in a loop, the screens can be swapped some 30 times per second. Pretty impressive for BASIC.

For even greater convenience, you can use the following statement, which displays your help screen until a key is pressed, then switches back to the normal screen:

POKE -16299,0:GET G\$:POKE -16300,0

Other uses for BLOADED screens include interactive software demonstrations and adventure games that show a screen and offer several options, each leading to another screen.

Designing a Help Screen

Any BASIC program that uses text page 2 must reserve space for that memory with POKE 104,12 and POKE 3072,0. This is the purpose of Program 1, which changes the bottom of program memory and calls Program 2, Help Screen Editor. (Don't confuse this process with changing the value of LOMEM, which merely relocates the bottom of variable memory, not program memory.) If you want, you can save Program 1 on a new disk with the filename HELLO so that it automatically boots the editor when the computer is turned on.

Type in and save Program 2 (make sure to use the filename SE if you want the program to work properly with Program 1). It's quite short for a full-featured editor.

The first action of the editor is to try to load a help screen for itself. Obviously, the very first time you run Help Screen Editor there will be no help screen for the editor on the disk for it to load. Your first project should be to create a help screen for the screen editor itself, so temporarily modify the program to skip loading a help screen by inserting this line:

15 GOTO 40

The help screen you make for Help Screen Editor should look something like the example in the accompanying screen photo.

Figure 1. Typical Help Screen



"Help Screen Editor" lets you add custom help screens to your own programs. This sample screen was created for use within the editor program itself.

Here are the screen editor commands:

Key	Function
Ctrl-I	Cursor up
Ctrl-J	Cursor left (or use the cursor-left key)
Ctrl-K	Cursor right (or use the cursor-right key)
Ctrl-M	Cursor down
Ctrl-N	Normal text
Ctrl-R	Reversed text
Ctrl-F	Flashing text
Ctrl-S	Save screen
Ctrl-L	Load screen
Ctrl-C	Clear screen
Ctrl-P	Print screen
Ctrl-T	Change title
Ctrl-Q	Quit editor
Esc	View help screen

Once you've created a help screen, save it to disk by pressing Ctrl-S and specifying a filename. For the screen editor's help screen, use the filename SEHELP (this filename is already in place in the program—see line 20 of Program 2). The program automatically appends the filename extender .SCR when saving or loading screen files. Delete the temporary line 15 added above, and you're ready to go. From now on, you can call up this help screen of screen editor commands merely by pressing the Esc key. Press any key to switch from the help screen back to the editor.

Programming Notes

To add help screens to your own programs, follow these steps:

1. Be sure your program reserves space for text page 2, just as Program 1 does, with POKE 104,12 and POKE 3072,0.
2. To load the help screen into memory, your program should execute the command `BLOAD filename,A$800`. Make sure the filename corresponds with the name of the screen file on the disk.
3. To swap text page 2 with text page 1 and make the help screen visible, your program should execute the statement `POKE -16299,0`. To return to the original screen, your program should execute the statement `POKE -16300,0`. If you want to make the help screen visible until any key is pressed, use `POKE -16299,0:GET G$:POKE -16300,0`.

Program 1. Screen Editor Loader

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
74 10 HOME : POKE 104,12: POKE 3072,0: PRINT CHR$ (4
    ); "RUN SE"
```

Program 2. Help Screen Editor

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
9E 10 HOME : R = 1: C = 1
74 20 PRINT CHR$ (4); "BLOAD SEHELP.SCR,A$800"
4A 30 POKE - 16299,0: GET G$: POKE - 16300,0
62 40 ST$ = "NEW SCREEN"
DC 50 DIM LS(23): FOR I = 1 TO 23: READ LS(I): NEXT
    : DATA 1024,1152,1280,1408,1536,1664,1792,1920
    ,1064,1192,1320,1448,1576,1704,1832,1960,1104,
    1232,1360,1488,1616,1744,1872
89 60 DIM CV(3): FOR I = 1 TO 3: READ CV(I): NEXT :
    DATA 3,6,27
57 70 D$ = CHR$ (4)
56 80 GOSUB 590
DE 90 VTAB R: HTAB C: GET G$: G = ASC (G$): IF G > 31
    THEN 500
83 100 FOR I = 1 TO 3: IF G = CV(I) THEN 120
88 110 NEXT I: GOTO 130
05 120 ON I GOSUB 150,170,180: GOTO 90
49 130 V = G - 7: IF V < 1 OR V > 14 THEN 90
E6 140 ON V GOSUB 190,210,230,240,260,300,320,330,34
    0,430,470,480,510,550: GOTO 90
32 150 VTAB 24: HTAB 1: PRINT "CLEAR THE SCREEN? (Y/
    N)";: GET G$: IF G$ = "Y" THEN HOME
A6 160 GOSUB 590: RETURN
DB 170 M = 2: GOSUB 590: FLASH : RETURN
76 180 POKE - 16299,0: GET G$: POKE - 16300,0: RETUR
    N
77 190 C = C - 1: IF C = 0 THEN C = 40: R = R - 1: IF
    R = 0 THEN R = 23: C = 40
12 200 RETURN
83 210 R = R - 1: IF R = 0 THEN R = 23
16 220 RETURN
90 230 GOSUB 190: RETURN
98 240 C = C + 1: IF C = 41 THEN C = 1: R = R + 1: IF
    R = 24 THEN R = 1: C = 1
1C 250 RETURN
17 260 NORMAL : M = 0: GOSUB 580: VTAB 24: HTAB 1: PO
    KE 34,23: PRINT "TITLE TO LOAD? <RET> = QUIT
    ";: INPUT "": T$
```

```

E6 270 IF LEN (T$) < 1 THEN 290
CB 280 ST$ = T$: PRINT D$"BLOAD ";ST$;".SCR"
B3 290 POKE 34,0: GOSUB 580: GOSUB 590: RETURN
6A 300 R = R + 1: IF R = 24 THEN R = 1
15 310 RETURN
10 320 M = 0: GOSUB 590: NORMAL : RETURN
19 330 RETURN
D6 340 POKE 34,23: VTAB 24: HTAB 1: PRINT "PRESS <RE
T> TO PRINT, OTHER TO ABORT.";
A5 350 GET G$:G = ASC (G$): IF G = 13 THEN 370
F0 360 PRINT CHR$ (4): POKE 34,0: GOSUB 580: GOSUB 5
90: RETURN
AF 370 GOSUB 580: POKE 34,23
66 380 PR# 1: FOR I = 1 TO 23: FOR J = 0 TO 39:P = P
EEK (LS(I) + J)
18 390 IF P < 192 THEN P = P + 64: GOTO 390
86 400 P = P - 128: IF P > 94 THEN P = P - 64
98 410 PRINT CHR$ (P);
82 420 NEXT J: PRINT : NEXT I: PRINT CHR$ (4): GOTO
360
D3 430 GOSUB 580: POKE 34,23: VTAB 24: HTAB 1: PRINT
"QUIT THE EDITOR? (Y/N)";
F1 440 GET G$: IF G$ = "Y" THEN POKE 34,0: HOME : EN
D
66 450 IF G$ = "N" THEN GOSUB 580: GOSUB 590: RETURN
1F 460 GOTO 440
C9 470 M = 1: GOSUB 590: INVERSE : RETURN
23 480 IF LEN (ST$) < 1 OR ST$ = "NEW SCREEN" THEN G
OSUB 510
27 490 NORMAL : GOSUB 580: VTAB 24: HTAB 1: PRINT "S
AVE ";ST$;"? (Y/N)";: GET G$: IF G$ = "Y" THE
N GOSUB 580: PRINT D$"BSAVE ";ST$;".SCR,A$400
,L$400"
AF 500 GOSUB 580: GOSUB 590: RETURN
82 510 NORMAL : GOSUB 580: HTAB 1: VTAB 24: PRINT "T
ITLE? ( <= 10 CHAR ) " CHR$ (4);: POKE 34,23:
INPUT T$: IF LEN (T$) > 10 THEN PRINT CHR$ (
7): GOTO 510
C3 520 IF LEN (T$) < 1 THEN 540
DE 530 ST$ = T$
85 540 GOSUB 580: POKE 34,0: GOSUB 590:R = 1:C = 1:M
= 0: RETURN
9B 550 GOSUB 240: RETURN
9D 560 PRINT G$;:C = C + 1: IF C > 40 THEN C = 1:R =
R + 1: IF R > 23 THEN R = 1
3F 570 GOTO 90
6E 580 POKE 34,23: VTAB 24: HTAB 1: PRINT :R = 1:C =
1: POKE 34,0: RETURN
03 590 NORMAL : VTAB 24: HTAB 1: ON M + 1 GOSUB 610,
620,630

```

```

CA 600 PRINT " EDITING ";: INVERSE : PRINT ST$:: NOR
MAL : VTAB 1: HTAB 1: PRINT CHR$ ( PEEK (1024
)): RETURN
66 610 PRINT "NORMAL ";: RETURN
D3 620 PRINT "REVERSED";: RETURN
C3 630 PRINT "FLASH ";: RETURN

```


MultiMemory

Patrick Parrish

"MultiMemory" is a pretty amazing program. Even though it's short, this utility partitions free memory so that several BASIC programs can be loaded into your computer at once. Among other things, it's a great aid during program development—you can keep a couple of BASIC programming utilities at hand as you work or test alternative versions of new routines before adding them to your main program. Works on all Apple II series computers using either DOS 3.3 or ProDOS.

The idea of partitioning memory into several modules which can contain separate programs is not new—programs written for other computers have handled the same problem. Like these earlier programs, "MultiMemory" divides free memory in your Apple into independent workspaces—four actually.

With these four partitions available, you can load different BASIC programs—utilities, applications, or games—into the computer at the same time. And you can save and load programs from any of these areas without affecting the others.

MultiMemory goes even further. Not only are the BASIC programs in each module protected from one another, but the variables generated by each are protected as well. Any program can change a variable's value without affecting identically named variables that may exist in other partitions. This means there's even less chance of conflict between the programs, allowing you more flexibility when using MultiMemory.

Entering MultiMemory

MultiMemory uses a section of memory called *zero page* (locations 0–255), the page's pointers in particular, to split up free memory. Locations 103–116 contain the pointers.

MultiMemory consists of short machine language routines entered by a BASIC loader. Carefully type in the program and save a copy to disk before running it for the first time.

When you run MultiMemory, line 100 sets location 8192 as the top of BASIC memory for the first partition. Line 110 POKes the machine language routine in lines 150–330 into a safe place in memory. It resides at location 38251, just below DOS (38400), an area relatively safe from memory conflicts.

Line 120 checks to see whether the machine language data has been correctly typed. Lines 130 and 140 place zeros in three sequential locations at the start of each memory module. The first zero is required to indicate the start of BASIC. The second and third zeros NEW each memory module. Finally, the NEW command in line 140 clears the BASIC loader from memory and leaves us in module 1.

Four Computers

To access any module, type CALL 38251 and press Return. No display or message appears, but you're ready to choose one of the four work areas. Press the 1, 2, 3, or 4 key, and a tone will sound, indicating that a partition has been picked. Again, for a test, specify number 2. A tone sounds, the partition number is displayed, and you're ready to program.

When you type LIST, you'll see that module 2 is empty. Now type PRINT FRE(0) to determine how much memory is available in this module. There should be about 8K free. That's plenty of room for a short BASIC program and its variables. To see that both a program and its variables remain intact within a particular module, let's enter and run a program in module 2 and then do the same in module 1.

While in module 2, type and run this program:

```
10 REM EXAMPLE 2
20 A$="MODULE #2"
30 FOR I=1 TO 20:NEXT I
```

After running this program, type PRINT A\$,I and press Return. You should see this:

```
MODULE #2      21
```

Save this program to disk with the filename P2.

Now go to another module. Before you do this, though, list the program in module 2 so that it's at the bottom of the screen. Now type CALL 38251 and choose module 1 by pressing the 1 key.

Independent Variables

After entering module 1, type LIST to prove that our first program has been left behind in module 2. Again, if you wish, type PRINT FRE(0) to determine the amount of memory available in this module. It should show approximately 6K.

Program P2 should still be on the screen. Even though it

was listed in module 1, it remains visible if you switch partitions without clearing the screen. This makes it possible to copy program lines from one module to another with the screen editor. Simply use the screen editing keys to cursor up to line 10. Change the 2 in this line to 1, cursor to the end of the line, and press Return to enter this line in memory. Line 10 is now copied into memory in module 1 *without disturbing line 10 in module 2*.

If the BASIC screen prompts don't obscure the other program lines, you can copy them to module 1 in the same manner. At any rate, lines 20 and 30 should read like this:

```
20 A$="MODULE #1"  
30 FOR I=1 TO 10:NEXT I
```

As you did before, run the program and then type PRINT A\$,I. The result is

```
MODULE #1      11
```

Save this program as P1.

Now, go back to module 2 with a CALL 38251 and type LIST. You should see program P2 on the screen unchanged. Print the values for A\$ and I. You'll see they still have the values they had when you left module 2.

Applications

As you can imagine, this process can be valuable if you're writing and debugging your own programs. Suppose you're writing a program in module 1 and you need a subroutine from a BASIC program you have on disk. Maybe you aren't sure which disk the program is on.

With MultiMemory, you can nimbly jump to another module, load the directory by entering CATALOG or CAT (for ProDOS), find the program you need, and then load it into that module or the third module, leaving the directory intact. Once you've found the subroutine you need from your earlier program, you can list it on the screen, shift back to the first partition, and copy the lines by Returning over them.

Suppose some bug in your program keeps the subroutine from working as you expected. Since variables retain their values within each program module, you can test each routine separately, compare the resulting variables, and make changes to your working version where needed.

With all this jumping from module to module, you might lose track of which partition you're in. To find out, just type `PRINT PEEK(38339)`.

In addition to aiding program development, MultiMemory can be used to hold a few BASIC programs that can be run individually. For instance, you might have a series of BASIC programs that, in sequence, manipulate data stored on disk. The first program could read in the data, manipulate it, and then write the results back to disk. In turn, a second, third, or fourth program stored in the other workspaces could do the same. Or you could designate one BASIC workspace for data storage, much like a RAM disk.

Before undertaking any sophisticated programming applications with MultiMemory, however, you should consider how it works and keep in mind a few restrictions on its use.

Memory Ceilings

As mentioned earlier, MultiMemory uses a series of BASIC pointers in zero page to set up each workspace. The values required by these pointers for each module are stored in a lookup table at the end of the program. Whenever you `CALL` MultiMemory, it stores the current zero-page values in positions within this table which correspond to the current module. The program then waits for you to specify the next module.

When you pick a module, MultiMemory reads the zero-page pointer values for the module and places them in their proper zero page locations. The pointers transferred are the starting addresses for the location of the BASIC program, the array and nonarray variables, the string variables, and the top of BASIC memory.

The barriers separating the partitions were selected to keep MultiMemory compatible with most programs. The first module runs from memory locations 2048-8191; the second, from 8192-16383; the third, from 16384-27317; and the fourth, from 27318-38250.

Note that the location of module 2 coincides with the Apple's first high-resolution graphics page. That means any programs which use the hi-res graphics page should be loaded into another module. And, of course, any programs loaded into module 2 will likely be erased if a program in another module uses the hi-res page.

MultiMemory

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```

B6 100 HIMEM: 8192
B3 110 FOR I = 38251 TO 38399: READ A: POKE I,A:X =
    X + A: NEXT
6B 120 IF X < > 16759 THEN PRINT "ERROR IN DATA STAT
    EMENTS.": STOP
5F 130 POKE 8192,0: POKE 8193,0: POKE 8194,0
B6 140 POKE 16384,0: POKE 16385,0: POKE 16386,0: POK
    E 27318,0: POKE 27319,0: POKE 27320,0: NEW
B3 150 DATA 174,195,149,189,195,149,170,160
C6 160 DATA 0,185,103,0,157,200,149,232
2F 170 DATA 200,192,14,208,244,173,0,192
52 180 DATA 201,128,144,249,141,16,192,170
A0 190 DATA 41,15,240,241,201,5,176,237
14 200 DATA 72,32,58,255,169,163,32,237
C0 210 DATA 253,138,32,237,253,32,142,253
B0 220 DATA 104,170,142,195,149,189,195,149
31 230 DATA 170,160,0,189,200,149,153,103
49 240 DATA 0,232,200,192,14,208,244,165
C7 250 DATA 105,133,175,165,106,133,176,96
70 260 DATA 1,0,14,28,42,1,8,3
2F 270 DATA 8,3,8,3,8,0,32,0
47 280 DATA 32,0,32,1,32,3,32,3
6E 290 DATA 32,3,32,0,64,0,64,0
01 300 DATA 64,1,64,3,64,3,64,3
BE 310 DATA 64,182,106,182,106,182,106,183
50 320 DATA 106,185,106,185,106,185,106,107
09 330 DATA 149,107,149,107,149

```

Apple Disk Duper

Jason Coleman

Duplicate disks quickly and conveniently. Though "Disk Duper" can copy disks formatted for either DOS 3.3 or ProDOS, it must be run under ProDOS. It also requires 128K RAM.

Everyone knows the value of backing up disks. But how many of us take the time to make archive copies of important disks on a regular basis? "Apple Disk Duper" simplifies the process by making it possible to copy an entire disk in only two passes. It works on single- or dual-drive systems with at least 128K RAM.

After typing in the program and saving a copy, simply run it and follow the instructions on the screen. Apple Disk Duper prompts you every step of the way.

Although the program runs only under ProDOS, it can copy DOS 3.3 disks as well as ProDOS disks. It works with any Apple Disk II-compatible drive, except the new 3½-inch UniDisk.

Apple Disk Duper

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
74 100 FOR X = 768 TO 785: READ Y: POKE X,Y: NEXT
AA 110 DATA 32,0,191,129,9,3,176,249,96,3,96,0,32,0,
      0,0,0,0,
5B 120 TEXT : HOME
47 130 VTAB 12: HTAB 12: PRINT "DISK DUPLICATOR"
AA 140 VTAB 20: HTAB 9: PRINT "(HIT ANY KEY TO BEGIN
      )"; POKE - 16368,0: GET ST$
4E 150 HOME
1C 160 VTAB 12: INPUT "ENTER NUMBER OF DRIVES:";ND$:
      ND = VAL (ND$)
CB 170 IF ND < > 1 AND ND < > 2 THEN 390
FI 180 HOME : VTAB 12: PRINT "PUT SOURCE DISK IN DRI
      VE 1"
7A 190 IF ND = 2 THEN VTAB 17: PRINT "PUT DESTINATIO
      N DISK IN DRIVE2"
D0 200 VTAB 20: POKE - 16368,0: PRINT "PRESS ANY KEY
      TO MAKE COPY.": GET AK$
90 210 FB = 0:MX = 3
91 220 FOR N = 1 TO MX
C4 230 POKE 771,128
11 240 POKE 780,32: POKE 778,96
```



```
1E 250 FOR I = FB TO FB + 55
25 260 P2 = INT (I / 256):P1 = I - 256 * P2
DE 270 POKE 782,P2: POKE 781,P1
50 280 CALL 768: POKE 780, PEEK (780) + 2: NEXT I
84 290 IF N < MX THEN PRINT CHR$ (4)"BSAVE/RAM/COPY"
    N",A$2000,L$6FFF":FB = FB + 56
62 300 NEXT N
28 310 IF ND = 1 THEN VTAB 12: PRINT "PUT DESTINATIO
    N DISK IN DRIVE 1": GET AK$
ED 320 FOR N = MX TO 1 STEP - 1
FB 330 POKE 771,129: POKE 780,142
15 340 IF ND = 2 THEN POKE 778,224
68 350 IF N < MX THEN PRINT CHR$ (4)"BLOAD/RAM/COPY"
    ;N
79 360 FOR I = FB + 55 TO FB STEP - 1:P2 = INT (I /
    256):P1 = I - 256 * P2
DF 370 POKE 782,P2: POKE 781,P1
71 380 CALL 768: POKE 780, PEEK (780) - 2: NEXT I
08 390 FB = FB - 56
63 400 NEXT N
18 410 IF MX = 2 THEN 440
60 420 MX = 2:FB = 168: IF ND = 1 THEN VTAB 12: PRIN
    T "PUT SOURCE DISK IN DRIVE 1      ": GET AK$
16 430 GOTO 220
07 440 HOME : VTAB 12: HTAB 15: INVERSE : PRINT "COP
    Y COMPLETE": NORMAL : END
```

Applesoft List Enhancer

Steven Roth

Tired of program listings that whiz across the screen at unreadable speeds? This short utility lets you scroll through a listing at your own pace, one screenful at a time. For all Apple II series computers using either DOS 3.3 or ProDOS.

LIST is one of the most frequently used commands in Applesoft BASIC, yet it has some inconvenient features. The program listing scrolls upward too fast to read, unless you repeatedly press Ctrl-S (Control and the S key). If you pass the program lines you want to see, you have to LIST the program again.

"Applesoft List Enhancer" is a machine language program that solves both of these problems. It divides the program listing into pages, and each page consists of one screenful of the listing. Instead of scrolling through the listing in only one direction, you now page through it, either forward or backward.

Type in and save the program below, then run it. This BASIC program automatically creates a machine language file on disk named ALE. (Don't use the name ALE for the BASIC program itself or you'll get a FILE TYPE MISMATCH error.) Once that's done, you don't need the BASIC program again except to create new copies of ALE.

To install and activate the List enhancer, type

BRUN ALE

Load any Applesoft BASIC program. To list it, type an ampersand (&) followed by an optional starting line number. If you don't enter any line number, the program is listed from the beginning. *(You can access only one program which uses the ampersand at a time. Thus, you can't use the List Enhancer at the same time as the "BASIC Line Editor" or "Apple Keyboard Customizer," also in this book.)*

The right- and left-arrow keys let you page forward or backward through the listing. Press the right-arrow key to display the next screenful of program lines; the left-arrow key

displays the previous page. To exit listing mode so that you can edit a line, press either the Esc key or Ctrl-C.

Applesoft List Enhancer

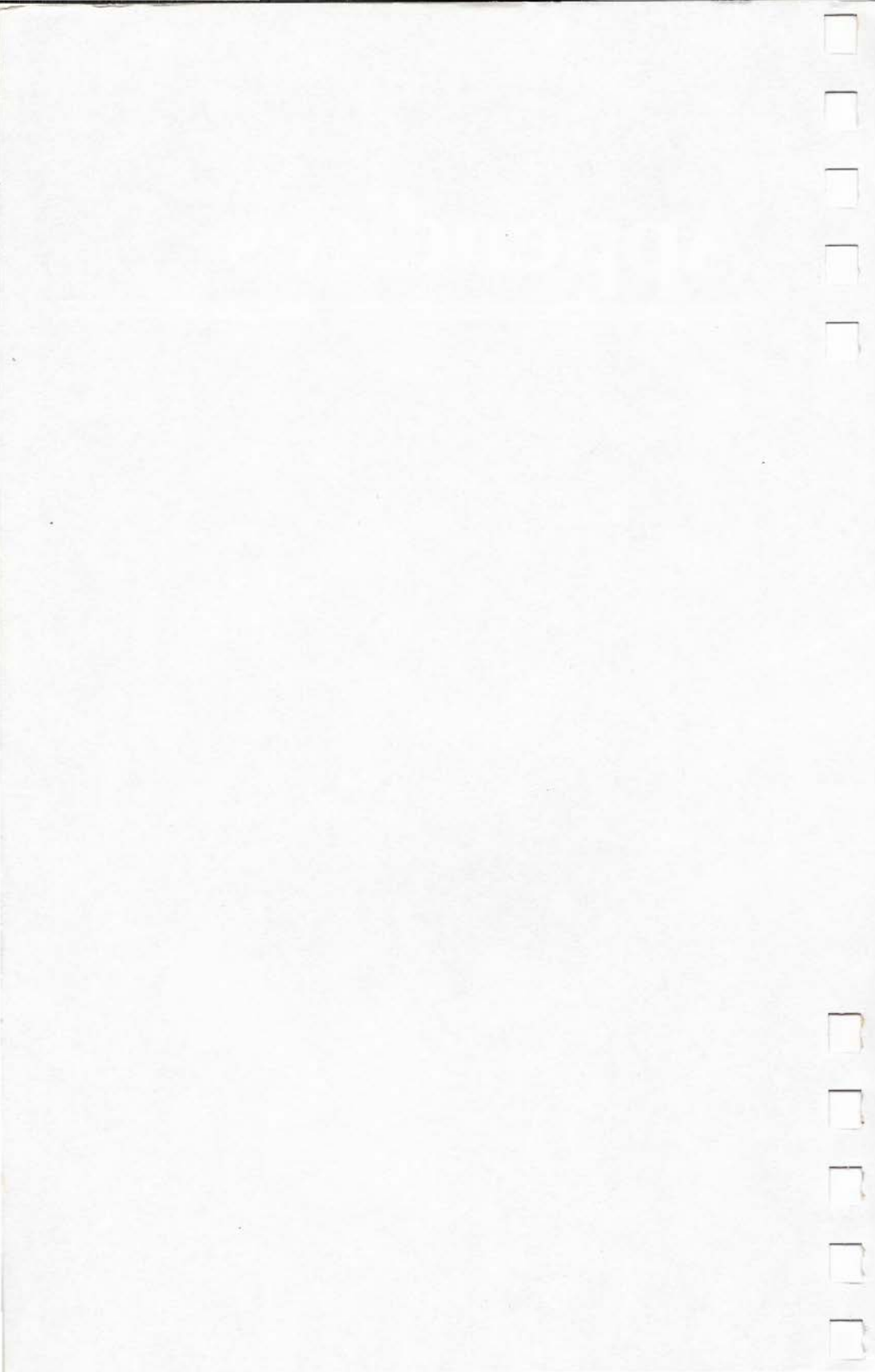
For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```

10 100 FOR I = 24576 TO 24847: READ A: POKE I,A:CK =
    CK + A: NEXT
10 110 IF CK < > 33557 THEN PRINT "ERROR IN DATA STA
    TEMENTS.": STOP
40 120 PRINT CHR$ (4)"BSAVE ALE,A$60000,L$110"
20 130 DATA 169,32,141,245,3,169,16,141
52 140 DATA 246,3,169,96,141,247,3,96
09 150 DATA 32,12,218,32,26,214,165,80
71 160 DATA 141,32,97,165,81,141,33,97
24 170 DATA 32,88,252,160,0,132,6,200
66 180 DATA 132,30,132,249,177,155,240,75
8A 190 DATA 200,177,155,170,200,177,155,132
00 200 DATA 9,133,8,134,7,32,36,237
1C 210 DATA 164,9,169,32,32,92,219,165
E0 220 DATA 36,201,33,144,15,230,30,165
82 230 DATA 30,201,23,240,48,32,142,253
61 240 DATA 169,5,133,36,200,177,155,208
5E 250 DATA 81,168,177,155,170,200,177,155
50 260 DATA 134,155,133,156,32,142,253,230
A1 270 DATA 30,165,30,201,23,240,14,160
08 280 DATA 1,208,177,230,6,230,6,169
60 290 DATA 0,133,249,240,17,230,6,230
68 300 DATA 6,164,6,165,7,153,32,97
45 310 DATA 200,165,8,153,32,97,32,12
0B 320 DATA 253,201,155,240,18,201,131,240
A0 330 DATA 14,201,149,240,101,201,136,240
17 340 DATA 58,32,58,255,76,150,96,76
A4 350 DATA 208,3,16,144,56,233,127,170
CA 360 DATA 132,9,160,208,132,157,160,207
78 370 DATA 132,158,160,255,202,240,7,32
37 380 DATA 44,215,16,251,48,246,169,32
B2 390 DATA 32,92,219,32,44,215,48,5
17 400 DATA 32,92,219,208,246,32,92,219
25 410 DATA 76,64,96,165,6,201,2,240
5F 420 DATA 192,56,233,4,133,6,166,6
C7 430 DATA 189,32,97,133,80,232,189,32
09 440 DATA 97,133,81,32,26,214,32,88
97 450 DATA 252,160,1,132,30,132,249,76
CA 460 DATA 44,96,165,249,208,224,240,153

```


Appendices



Guide to Typing In Programs

What Is a Program?

A computer cannot perform any task by itself. Like a car without gas, a computer has *potential*, but without a program, it isn't going anywhere. Most of the programs in this book are written in a computer language called Applesoft BASIC. It's easy to learn and works on the Apple II, II+, IIe, and IIfx.

BASIC Programs

Computers can be picky. Unlike the English language, which is full of ambiguities, BASIC usually has only one right way of stating something. Every letter, character, and number is significant. A common mistake is substituting a letter such as *O* for the numeral 0, a lowercase *l* for the numeral 1, or an uppercase *B* for the numeral 8. Also, you must enter all punctuation marks such as colons and commas just as they appear in the listings. Spacing can be important. To be safe, type in the programs exactly as they appear. Unlike other program listings you may have seen, those in this book have no special characters which you need to interpret. Simply enter the programs as they appear.

DOS 3.3 and ProDOS

Unless otherwise mentioned in the program's accompanying article, it doesn't matter whether you have DOS 3.3 or ProDOS. You can enter the programs with either DOS active in your Apple. Of course, you can run a typed-in program only with the DOS system it was entered with.

Uppercase

You'll notice that all the program listings are entirely in uppercase. If you have an Apple IIe or IIfx, however, which allows both uppercase and lowercase, you can change text which appears in PRINT statements if you want. A program such as "Your Personal Ledger," for instance, could be modified so that the screen displays appear in both uppercase and lowercase.

DATA Statements

Some programs contain one or more sections of DATA statements. These lines provide information needed by the program. Some DATA statements contain actual programs (called machine language); others contain graphics codes. These lines are especially sensitive to errors.

If a single number in any one DATA statement is mistyped, your machine could lock up, or crash. The keyboard may seem dead, and the screen may go blank. Don't panic—no damage is done. To regain control, you have to turn off your computer, then turn it back on. This will erase whatever program was in memory, *so always save a copy of your program before you run it*. If your computer crashes, you can load the program and look for your mistake.

Sometimes, a mistyped DATA statement will cause an error message when the program is run. The error message may refer to the program line that READs the data. *The error is still in the DATA statements, though.*

Get to Know Your Machine

You should familiarize yourself with your computer before attempting to type in a program. Learn the statements you use to store and retrieve programs from tape or disk. You'll want to save a copy of your program so that you won't have to type it in every time you use it. Learn to use your machine's editing functions. How do you change a line if you make a mistake? You can always retype the line, but you at least need to know how to use the left- and right-arrow keys. It's all explained in your computer's manuals.

A Quick Review

1. Type in the program, a line at a time, in order. Press Return at the end of each line. Use the left arrow to correct mistakes.
2. Check the line you've typed against the line in the book. You can check the entire program again if you get an error when you run the program.

Apple Automatic Proofreader

Tim Victor

It's easier than ever to enjoy programs for Apple II series computers. "Apple Automatic Proofreader," an error-checking program for the Apple II, II+, IIe, and IIC with either DOS 3.3 or ProDOS, alerts you to almost every typing mistake you might make.

"Apple Automatic Proofreader" will help you type in program listings without typing mistakes. It's a short error-checking program that hides itself in memory and attaches to your Apple's operating system. Each time you press Return to enter a program line, this routine displays a two-digit checksum at the top of your screen. If you've typed the line correctly, the checksum on your screen matches the one in the printed listing—it's that simple. You don't have to use the Proofreader to enter listings, but doing so greatly reduces the chance of making a typo.

Getting Started

First, type in the Apple Automatic Proofreader program following this article. The Proofreader can't check itself before it's done, so you'll have to be extra careful to avoid mistakes.

The Proofreader checks which operating system you're running before it hooks up the checksum routine, so you can type it in with either DOS 3.3 or ProDOS. If you want to use the Proofreader with both operating systems, you won't have to retype it. All you need is a utility to copy a file between disks with different formats, such as the one provided on the ProDOS *System Utilities* disk.

As soon as you finish typing the Proofreader, save at least two copies. This is very important, because the Proofreader erases the BASIC portion of itself when you run it, leaving only the machine language portion in memory.

Now, type RUN and hit Return. The Proofreader clears the screen, loads the machine language routine, displays the message PROOFREADER ACTIVATED, erases the BASIC portion of itself, and ends. If you type LIST and press Return,

you'll see that no BASIC program is in memory. The computer is ready for you to type in a new BASIC program.

Entering Programs

Once the Proofreader is activated, you can begin typing in a BASIC program as usual. Every time you finish typing a line and press Return, the Proofreader displays a two-digit checksum number in the upper-left corner of the screen. Compare this checksum with the two-digit checksum printed next to the corresponding line in the program listing. If the numbers match, you can be pretty certain the line was typed correctly. Otherwise, check for your mistake and type the line again.

A common mistake when entering BASIC programs on the Apple occurs when you accidentally press a key while holding down the Ctrl (Control) key. This adds an invisible control character to the line you are typing. If you don't find it before you run the program, this stray character may cause a SYNTAX ERROR or other mysterious behavior. Fortunately, the Proofreader detects the presence of these invisible control characters, displaying a checksum that doesn't match the one in the listing. So it's always a good idea to retype a line if the checksums don't match, even though you might not see any difference in the lines themselves.

The Proofreader ignores space characters, so you can omit spaces between keywords and still see a matching checksum. Spaces are important only between the quotation marks of PRINT statements or string assignments. If you accidentally type too many spaces or leave some out, this is the only mistake the Proofreader won't catch. For this reason, you should be extra careful when entering text within quotation marks.

Before running another BASIC program, it's a good idea to turn off the Proofreader by holding down the Ctrl key while pressing the Reset button. The machine language part of the Proofreader is kept in memory starting at address 768 (\$300 hexadecimal). This location is out of BASIC's way, but a lot of other programs use this same place for their machine language subroutines. Disable the Proofreader to avoid conflicts.

How It Works

When the Applesoft BASIC interpreter needs to get a line of

input from the keyboard, it calls a machine language routine in the Apple's read only memory (ROM) called GETLN. GETLN, in turn, calls the operating system to get a single keypress, which it stores in an input buffer. If the Return key was pressed, GETLN ends, leaving one new line for the BASIC interpreter in the input buffer. Otherwise, it repeats the process, asking for another keypress.

The operating system normally gets individual keystrokes from a ROM routine called KEYIN, but the Proofreader changes this. When the Proofreader is installed, the operating system calls the checksum routine instead, and the checksum routine asks KEYIN for a character. If any key other than Return was pressed, the checksum routine just passes it on to the operating system, which gives it to GETLN. But if Return *was* pressed, the checksum routine examines the contents of GETLN's input buffer, which now contains an entire line of input, to calculate the checksum that it displays at the top of the screen.

One very common typing mistake is transposition—typing two successive characters in the wrong order, like *PIRNT* instead of *PRINT*. A checksum program that merely adds the codes of the characters in a line can detect only the presence or absence of a character, not transposition errors. Because the Apple Proofreader uses a sophisticated formula to compute checksums, it alerts you to transposed keystrokes.

The Apple Automatic Proofreader detects almost every possible typing mistake, including transpositions, missing or extra characters, accidental control characters, and incorrect line numbers. Typing *COMPUTE!* Publications' programs into your Apple computer has never been easier.

Apple Automatic Proofreader

```

52 100 C = 0: FOR I = 768 TO 768 + 68: READ A: C = C +
      A: POKE I, A: NEXT
80 200 IF C < > 7258 THEN PRINT "ERROR IN PROOFREADER
      DATA STATEMENTS": END
00 300 IF PEEK (190 * 256) < > 76 THEN POKE 56, 0: POK
      E 57, 3: CALL 1002: GOTO 50
70 400 PRINT CHR$ (4); "IN#A$300"
24 500 POKE 34, 0: HOME : POKE 34, 1: VTAB 2: PRINT "PR
      OOFREADER INSTALLED"
FE 600 NEW
52 100 DATA 216, 32, 27, 253, 201, 141

```

```
10 110 DATA 208,60,138,72,169,0
75 120 DATA 72,189,255,1,201,160
FA 130 DATA 240,8,104,10,125,255
47 140 DATA 1,105,0,72,202,208
1B 150 DATA 238,104,170,41,15,9
AF 160 DATA 48,201,58,144,2,233
08 170 DATA 57,141,1,4,138,74
9E 180 DATA 74,74,74,41,15,9
B5 190 DATA 48,201,58,144,2,233
E3 200 DATA 57,141,0,4,104,170
A9 210 DATA 169,141,96
```

Apple MLX Machine Language Entry Program

Tim Victor

Machine language programs are difficult to enter into your computer. To make this chore easier and to eliminate typing mistakes, COMPUTE! Publications presents a machine language entry program for the Apple II, II+, IIe, and IIfx computers, using either the DOS 3.3 or ProDOS operating system.

A machine language program is usually listed as a long series of numbers. It's hard to keep your place and even harder to avoid making mistakes as you type in the listing since an incorrect line looks almost the same as a correct one. To reduce the problems associated with typing in machine language programs, we've presented them as MLX listings which can be entered using the "Apple MLX" editor.

MLX checks your typing on a line-by-line basis. It won't let you enter inappropriate characters, and it won't let you continue if there's a mistake in a line or even if you're trying to enter a line or digit out of sequence. You don't have to know anything about machine language to use it. In other words, MLX makes machine language program entry almost foolproof.

Using MLX

Type in and save MLX to disk (you'll want to use it to enter a number of programs in this book as well as some of those listed in *COMPUTE!* magazine and *COMPUTE!'s Apple Applications Special* issues). It doesn't matter whether you type it in on a disk formatted for DOS 3.3 or ProDOS. Programs entered with MLX, however, must be saved to a disk formatted with the same operating system as MLX itself.

If you have an Apple IIe or IIfx, make sure that the key marked Caps Lock is in the down position. Type RUN. You'll be asked for the starting and ending addresses of the machine language program. These values are given at the beginning of

the machine language program listing and in the program's accompanying article. Find them and type them in.

The next thing you'll see is a menu asking you to select a function. The first is (E)NTER DATA. If you're just starting to type in a program, pick this. Press the E key, and the program asks for the address where you want to begin entering data. Type the first number in the first line of the program listing if you're just starting or the line number where you left off if you've already typed in part of a program. Hit the Return key and begin entering the data.

Once you're in enter mode, MLX will print the address for each program line for you. You then type in all nine numbers on that line, beginning with the first two-digit number after the colon (:). Each line represents eight bytes and a checksum. When you enter a line and hit Return, MLX recalculates the checksum from the eight bytes and the address. If you enter more than or fewer than nine numbers, or if the checksum doesn't exactly match, MLX erases the line you just entered and prompts you again for the same line.

Invalid Characters Banned

MLX is fairly flexible about how you type in the numbers. You can put extra spaces between numbers or leave the spaces out entirely, compressing a line into 18 keypresses. Be careful not to put a space between two digits in the middle of a number. MLX will read two single-digit numbers instead of one two-digit number (F 6 means F and 6, not F6).

You can't enter an inappropriate character with MLX. Only the numerals 0-9 and the letters A-F can be typed in. If you press any other key (with some exceptions noted below), nothing happens. This safeguards against entering extraneous characters. Even better, MLX checks for transposed characters. If you're supposed to type in A0 and instead enter 0A, MLX will catch your mistake.

MLX also checks to make sure you're typing in the right line. The address (the number to the left of the colon) is part of the checksum recalculation. If you accidentally skip a line and try to enter incorrect values, MLX won't let you continue. Just make sure you enter the correct starting address; if you don't, you won't be able to enter any of the following lines. MLX will stop you.

Editing Features

MLX also includes some editing features. The left- and right-arrow keys allow you to back up and go forward on the line you're entering so that you can retype data. Pressing the Ctrl (Control) key and the D key at the same time (*Delete*) removes the character under the cursor, shortening the line by one character. Pressing the Ctrl key and the I key simultaneously (*Insert*) puts a space under the cursor and shifts the rest of the line to the right, making the line one character longer. If the cursor is at the right end of the line, neither Ctrl-D nor Ctrl-I has any effect.

When you've entered the entire listing (up to the ending address that you specified earlier), MLX automatically leaves enter mode and redisplay the functions menu. If you want to leave enter mode before then, press the Return key when MLX prompts you with the address of a new line.

Display Data

The second menu choice, (D)ISPLAY DATA, examines memory and shows the contents in the same format as the program listing. You can use it to check your work or to see how far you've got. When you press the D key, MLX asks you for a starting address. Type in the address of the first line that you want to see and hit Return. MLX displays program lines until you press any key or until it reaches the end of the program.

Save and Load

Two menu selections are provided to let you save programs to disk and load them back into the computer. These are (S)AVE FILE and (L)OAD FILE. MLX asks you for the name of the file which contains the program. The first time you save a machine language program, there won't be a file on the disk containing the program. Whatever name you type in will be the name of a new file that's created.

The message DISK ERROR appears during a save or load if a problem is detected. If you're not sure why a disk error has occurred, check the disk drive. Make sure there's a formatted disk in the drive and that it was formatted by the same operating system that you're using for MLX (ProDOS or DOS 3.3). If you're trying to save a file and see an error message, the disk might be full. Either save the file on another disk or quit MLX (by pressing Q), delete an old file or two, then run

MLX again. Your typing should still be safe in memory. If the error message appears during a load, you may have specified a filename that doesn't exist on the disk.

Quit

The Quit menu option has the obvious effect—it stops MLX and enters BASIC. (Of course, you can also press Ctrl-Reset to get out of MLX.)

The Finished Product

When you've finished typing all the data for a machine language program and saved your work, you're ready to see the results. The instructions for loading and using the finished product vary from program to program. You'll almost always load and run an MLX-generated program by typing `BRUN filename` (or sometimes just `BLOAD`).

An Ounce of Prevention

By the time you finish typing in the data for a long program, you may have several hours invested in the project. Don't take chances—use the "Apple Automatic Proofreader" to enter MLX, and then test your copy *thoroughly* before first using it to enter any significant amount of data. Make sure all the menu options work as they should. Enter fragments of the program starting at several different addresses, then use the Display option to verify that the data has been entered correctly. And be sure to test the Save and Load options several times to insure that you can recall your work from disk. Don't let a simple typing error in MLX cost you several nights of hard work.

Line 100 of MLX traps all errors to line 610. If MLX is typed in correctly, then only disk errors should be encountered. A disk error message when you're not trying to access the drive—for example, when you first start entering data—indicates a typing error in the MLX program itself. If this occurs, hit Ctrl-Reset to break out of MLX and carefully compare your entry against the printed listing.

Apple MLX: Machine Language Entry Program

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
00 100 N = 9: HOME : NORMAL : PRINT "APPLE MLX": POK
    E 34,2: ONEKR GOTO 610
```



```

CC 110 VTAB 1: HTAB 20: PRINT "START ADDRESS";: GOSU
B 530: IF A = 0 THEN PRINT CHR$ (7): GOTO 110
BC 120 S = A
EC 130 VTAB 2: HTAB 20: PRINT "END ADDRESS ";: GOSU
B 530: IF S >= A OR A = 0 THEN PRINT CHR$ (7
): GOTO 130
20 140 E = A
B5 150 PRINT : PRINT "CHOOSE:(E)NTER DATA";: HTAB 22
: PRINT "(D)ISPLAY DATA": HTAB 8: PRINT "(L)O
AD FILE (S)AVE FILE (Q)UIT": PRINT
AE 160 GET A$: FOR I = 1 TO 5: IF A$ < > MID$ ("EDLS
Q",I,1) THEN NEXT : GOTO 160
F3 170 ON I GOTO 270,220,180,200: POKE 34,0: END
AF 180 INPUT "FILENAME: ";A$: IF A$ < > "" THEN PRIN
T CHR$ (4);"BLOAD";A$;"A";S
A1 190 GOTO 150
6D 200 INPUT "FILENAME: ";A$: IF A$ < > "" THEN PRIN
T CHR$ (4);"BSAVE";A$;"A";S;"L";E - S
92 210 GOTO 150
C2 220 GOSUB 590: IF B = 0 THEN 150
9E 230 FOR B = B TO E STEP 8:L = 4:A = B: GOSUB 580:
PRINT A$;" ":L = 2
B5 240 FOR F = 0 TO 7:V(F + 1) = PEEK (B + F): NEXT
: GOSUB 560:V(9) = C
F2 250 FOR F = 1 TO N:A = V(F): GOSUB 580: PRINT A$"
":NEXT : PRINT : IF PEEK (49152) < 128 THE
N NEXT
94 260 POKE 49168,0: GOTO 150
CC 270 GOSUB 590: IF B = 0 THEN 150
48 280 FOR B = B TO E STEP 8
A6 290 HTAB 1:A = B:L = 4: GOSUB 580: PRINT A$;" ":
: CALL 64668:A$ = "" : P = 0: GOSUB 330: IF L =
0 THEN 150
F9 300 GOSUB 470: IF F < > N THEN PRINT CHR$ (7): G
OTO 290
27 310 IF N = 9 THEN GOSUB 560: IF C < > V(9) THEN P
RINT CHR$ (7): GOTO 290
72 320 FOR F = 1 TO 8: POKE B + F - 1,V(F): NEXT : P
RINT : NEXT : GOTO 150
8E 330 IF LEN (A$) = 33 THEN A$ = 0$:P = 0: PRINT CH
R$ (7);
22 340 L = LEN (A$):O$ = A$:O = P:L$ = "": IF P > 0
THEN L$ = LEFT$ (A$,P)
E0 350 R$ = "": IF P < L - 1 THEN R$ = RIGHT$ (A$,L
- P - 1)
55 360 HTAB 7: PRINT L$: FLASH : IF P < L THEN PRIN
T MID$ (A$,P + 1,1): NORMAL : PRINT R$;
7B 370 PRINT " ": NORMAL
E6 380 K = PEEK (49152): IF K < 128 THEN 380
C1 390 POKE 49168,0:K = K - 128

```

```

53 400 IF K = 13 THEN HTAB 7: PRINT A$; " ";: RETURN
8A 410 IF K = 32 OR K > 47 AND K < 58 OR K > 64 AND
    K < 71 THEN A$ = L$ + CHR$ (K) + R$: P = P + 1
C1 420 IF K = 4 THEN A$ = L$ + R$
5F 430 IF K = 9 THEN A$ = L$ + " " + MID$ (A$, P + 1,
    1) + R$
0A 440 IF K = 3 THEN P = P - (P > 0)
93 450 IF K = 21 THEN P = P + (P < L)
9D 460 GOTO 330
37 470 F = 1: D = 0: FOR P = 1 TO LEN (A$): C$ = MID$
    (A$, P, 1): IF F > N AND C$ < > " " THEN RETURN
E9 480 IF C$ < > " " THEN GOSUB 520: V(F) = J + 16 *
    (D = 1) * V(F): D = D + 1
5F 490 IF D > 0 AND C$ = " " OR D = 2 THEN D = 0: F =
    F + 1
0B 500 NEXT : IF D = 0 THEN F = F - 1
17 510 RETURN
55 520 J = ASC (C$): J = J - 48 - 7 * (J > 64): RETUR
    N
AB 530 A = 0: INPUT A$: A$ = LEFT$ (A$, 4): IF LEN (A$
    ) = 0 THEN RETURN
6F 540 FOR P = 1 TO LEN (A$): C$ = MID$ (A$, P, 1): IF
    C$ < "0" OR C$ > "9" AND C$ < "A" OR C$ > "Z"
    THEN A = 0: RETURN
2D 550 GOSUB 520: A = A * 16 + J: NEXT : RETURN
28 560 C = INT (B / 256): C = B - 256 * C - 255 * (C
    > 127): C = C - 255 * (C > 255)
20 570 FOR F = 1 TO 8: C = C * 2 - 255 * (C > 127) +
    V(F): C = C - 255 * (C > 255): NEXT : RETURN
0A 580 I = FRE (0): A$ = "": FOR I = 1 TO L: T = INT (
    A / 16): A$ = MID$ ("0123456789ABCDEF", A - 16
    * T + 1, 1) + A$: A = T: NEXT : RETURN
1F 590 PRINT "FROM ADDRESS ";: GOSUB 530: IF S > A 0
    R E < A OR A = 0 THEN B = 0: RETURN
0D 600 B = S + 8 * INT ((A - S) / 8): RETURN
8A 610 PRINT "DISK ERROR": GOTO 150

```

Disk Instructions

Typing in a long BASIC or machine language program can be a time-consuming task. Even with sophisticated error-checking programs like "Apple Automatic Proofreader" and "Apple MLX," you still have to spend hours in front of the computer.

That's why we've made available for purchase a disk containing all the programs in this book. To order the *COMPUTE!'s Third Book of Apple* companion disk, use the coupon in the back of this book, or call toll-free 1-800-346-6767 (in NY 1-212-887-8525).

Preparing a Purchased Disk

If you've bought *COMPUTE!'s Third Book of Apple* disk and simply put it in your computer's disk drive and turn the computer on, you'll see this message:

COMPUTE!'S THIRD BOOK OF APPLE

(C) COPYRIGHT 1986 ALL RIGHTS RESERVED

**INSERT A DOS 3.3 DISK THEN
PRESS ANY KEY TO BOOT**

or

**INSERT A PRODOS DISK THEN
PRESS ANY KEY TO BOOT**

The disk you purchased doesn't contain the version of DOS (Disk Operating System) necessary to start the Apple. You have two options. You can boot the system each time with a disk that does contain DOS 3.3 or ProDOS (look for a DOS 3.3 disk called *System Master* or a ProDOS disk called *ProDOS User's Disk*—one or the other came with your computer), then insert the *Third Book of Apple* disk and type **RUN MENU**.

A better way is to copy programs from the *Third Book of Apple* disk to your own disk, one that has DOS 3.3 or ProDOS already on it. It's not hard.

Note: *You'll need two disks to hold all the files on the Third Book of Apple disk in addition to the DOS 3.3 or ProDOS system files. Splitting the files between two disks means that you must only select from the MENU those programs copied to the specific disk. If you select a program which wasn't copied to that disk, you'll receive an error message.*

DOS 3.3 Users with Two Disk Drives

1. Insert your *System Master* disk in drive 1 and turn on the computer.
2. Insert the *Third Book of Apple* disk in drive 2. Be sure that the side labeled DOS 3.3 is facing upward.
3. Type **LOAD MENU,D2**, then press the Return key.
4. Remove the *Third Book of Apple* disk from drive 2, and insert a blank, unformatted disk in drive 2.
5. Type **INIT HELLO,D2**, then press Return. The disk in the drive will be formatted and the DOS system files will be written to the disk. When the formatting is completed, remove the disk from drive 2, place another blank, unformatted disk in drive 2, and type **INIT HELLO,D2** again. (You'll need two disks of your own to hold all the files on the *Third Book of Apple* disk in addition to the DOS system files.)
6. When the cursor reappears, type **BRUN FID,D1** and press Return.
7. Remove your *System Master* disk from drive 1, and insert the *Third Book of Apple* disk. Again, be sure that the side labeled DOS 3.3 is facing upward.
8. From the File Developer menu prompt, press 2, then Return, to select the CATALOG option. For the SOURCE SLOT? prompt, type 6, then Return, and for the DRIVE? prompt, type 1 and then Return.
9. Write down all the filenames (except MENU) from the *Third Book of Apple* disk, then press any key to get back to the File Developer menu.
10. Type 1, then press Return, to select the COPY FILES option. For the SOURCE SLOT? prompt, type 6, then Return, and for the DRIVE? prompt, type 1, then Return. For the DESTINATION SLOT? prompt, type 6, then Return, and for the DRIVE? prompt, type 2, then Return.
11. Type in the name of the file to be transferred, followed by Return, and hit any key to proceed with the copy. Then hit any key to return to the menu.
12. Type 2, then Return, if you need to refresh your memory about filenames (select slot 6, drive 1). Then hit any key for the menu.

13. Otherwise, repeat steps 10–12 until all files are copied. Slot and drive designations will default to your specifications unless changed, so you will not have to enter these again for each file. Change to the second formatted disk when you receive this message:

DISK FULL

CANCELLED

PRESS ANY KEY TO CONTINUE

14. Copy files from the *Third Book of Apple* disk to your second disk as you did before.
15. For the MENU to work, the files AASET and HROUT must be copied to both disks.

DOS 3.3 Users with One Disk Drive

1. Insert your *System Master* disk and turn on the computer.
2. Remove the *System Master* disk and insert the *Third Book of Apple* disk. Be sure that the side labeled DOS 3.3 is facing upward.
3. Type **LOAD MENU**, then press Return.
4. Remove the *Third Book of Apple* disk, and insert a blank, unformatted disk in the drive.
5. Type **INIT HELLO**, then press Return. The blank disk will be formatted and DOS system files will be written to the disk. When the formatting is completed, remove the disk from the drive, place another blank, unformatted disk in the drive, and type **INIT HELLO**, again. (You'll need two disks of your own to hold all the files on the *Third Book of Apple* disk plus the DOS system files.)
6. When the cursor reappears, remove the second newly formatted disk and insert your *System Master* disk.
7. Type **BRUN FID**, then Return.
8. Remove the *System Master* disk and insert the *Third Book of Apple* disk in the drive. Again, be sure that the side labeled DOS 3.3 is facing upward.
9. At the File Developer menu prompt, type 2, then press Return, for the CATALOG option. At the SOURCE SLOT? prompt, type 6, then Return, and for the DRIVE? prompt, type 1, then Return.
10. Write down all the filenames (except MENU) from the *Third Book of Apple* disk, then press any key to get back to the File Developer menu.

11. Type 1, then press Return, to select the COPY FILES option. At the SOURCE SLOT? prompt, type 6, then Return, and for the DRIVE? prompt, type 1, then Return. For the DESTINATION SLOT? prompt, type 6, Return, and for the DRIVE? prompt, type 1, Return.
12. Type in the filename of the first file to be transferred and press Return; then press any key *twice* to proceed with the transfer.
13. Remove the *Third Book of Apple* disk, insert your newly formatted disk, and press any key to make the copy.
14. Remove your disk and insert the *Third Book of Apple* disk again; then press any key to return to the File Developer menu.
15. Type 2, then press Return, if you need to see the list of filenames again (specify slot 6, drive 1).
16. Repeat steps 11–15 until all files are copied. Slot and drive designations will default to your specifications unless changed, so you will not have to enter these again for each file. As you swap disks back and forth, be very careful that you always insert the *Third Book of Apple* disk with the side labeled DOS 3.3 facing upward.
17. Change to the second formatted disk when you receive this message:
**DISK FULL
CANCELLED
PRESS ANY KEY TO CONTINUE**
18. Copy files from the *Third Book of Apple* disk to your second disk as you did before.
19. For the MENU program to work, the files AASET and HROUT must be copied to both disks.

ProDOS Users with Two Disk Drives

1. Put your *ProDOS User's Disk* in drive 1, and a blank, unformatted disk in drive 2, and turn on your computer.
2. From the master menu, type F for PRODOS FILER (UTILITIES). From the Filer menu, type V for VOLUME COMMANDS. From the Volume Commands menu, type F for FORMAT A VOLUME. Type in 6 for SLOT, 2 for DRIVE, and TBA1 as your NEW VOLUME NAME. (You can use another volume name if you want.) After you press Return, the formatting will begin. When you see the message FORMAT COMPLETE, press Esc twice to return to the

Filer menu. Remove the disk from drive 2, put another blank, unformatted disk in drive 2, and follow the above instructions again. This time, type **TBA2** as your NEW VOLUME NAME. (You'll need two disks of your own to hold all the files on the *Third Book of Apple* disk.) When the second disk is formatted, replace it with the first formatted disk.

3. From the Filer menu, type **F** for FILE COMMANDS, then **C** for COPY FILES. For the COPY PATHNAME prompt, type **/USERS.DISK/PRODOS** and press Return; for the TO PATHNAME prompt, type **/TBA1/PRODOS** and press Return. When you press Return again, the file PRODOS will be copied to your newly formatted disk. When you see the message COPY COMPLETE, press Return and copy the file BASIC.SYSTEM by typing **/USERS.DISK/BASIC.SYSTEM** for the COPY PATHNAME prompt; press Return. Then type **/TBA1/BASIC.SYSTEM** for the TO PATHNAME prompt and press Return. You can simply use the cursor to change the filename at the end of each prompt. Press Return to begin the copy. When you see the message COPY COMPLETE, press Esc. Follow the above instructions to copy the files PRODOS and BASIC.SYSTEM to your second disk (the one with the volume name TBA2).
4. When prompted, type **L** for LIST PRODOS DIRECTORY. Remove the *ProDOS User's Disk* from drive 1, and insert the *Third Book of Apple* disk. Be sure the side labeled ProDOS is facing upward. Now for the DIRECTORY PATHNAME prompt, type **/APP3BK** and press Return. When the directory appears, write down all the filenames, then press Return, and write down any other filenames that appear. Press Esc to return to the menu.
5. When prompted, press **C** for COPY FILES. For the COPY PATHNAME prompt, type **/AAP3BK/MENU** and press Return; for the TO PATHNAME prompt, type **/TBA1/MENU** and press Return. Press Return again, and the copy will begin. Repeat this procedure for each filename you wrote down earlier. Remember, you can cursor over the pathname and change only the filenames rather than typing in both each time.
6. When you see the message
FILE NAME TOO LARGE

it's time to switch disks in drive 2. Remove the TBA1 disk and replace it with the TBA2 disk. Continue with your copying, substituting TBA2 for TBA1 in the pathname. Be sure to copy MENU, AASET, and HROUT to both disks.

7. To use either disk, type **RUN MENU** at the] prompt and press Return.

ProDOS Users with One Disk Drive

1. Put your *ProDOS User's Disk* in the drive and turn on the computer.
2. From the master menu, type **F** for PRODOS FILER (UTILITIES). From the Filer menu, type **V** for VOLUME COMMANDS. From the Volume Commands menu, type **F** for FORMAT A VOLUME. Next, type **6** and press Return for the SLOT prompt, then type **1** and press Return for the DRIVE prompt. For the NEW VOLUME NAME prompt, type **TBA1** and press Return. (You can use something else if you want.) After you press Return, the formatting will begin. When you see the message **FORMAT COMPLETE**, press **Esc** twice to return to the Filer menu. Remove the disk from drive 2, put another blank, unformatted disk in drive 2, and follow the above instructions again. This time, type **TBA2** as your NEW VOLUME NAME. (You'll need two disks of your own to hold all the files on the *Third Book of Apple* disk.)
3. When you see the message **FORMAT COMPLETE** for the second time, remove the newly formatted disk and reinsert the *ProDOS User's Disk* into the drive. Then, press **Esc** twice to return to the Filer menu.
4. Type **F** for FILE COMMANDS, then **C** for COPY FILES. For the COPY PATHNAME prompt, type **/USERS.DISK/PRODOS** and press Return. For the TO PATHNAME prompt, type **/TBA1/PRODOS** and press Return. Press Return again, and the file PRODOS will be placed into memory. When prompted, remove the *ProDOS User's Disk* and insert your first formatted disk, then press Return. A copy of the file PRODOS will be placed on your newly formatted disk. When you see the **COPY COMPLETE** message, reinsert the *ProDOS User's Disk* and copy the file **BASIC.SYSTEM** by typing **/USERS.DISK/ BASIC.SYSTEM** and pressing Return for the COPY PATHNAME prompt. Type **/TBA1/BASIC.SYSTEM**, then press Return for the

TO PATHNAME prompt. Press Return again, and the file BASIC.SYSTEM will be placed into memory. When prompted, remove the *ProDOS User's Disk* and reinsert your newly formatted disk, then press Return. A copy of the file BASIC.SYSTEM will be placed onto your newly formatted disk. When you see the message COPY COMPLETE, press Esc to return to the File Commands menu. Follow the above instructions to copy the files PRODOS and BASIC.SYSTEM to your second disk (the one with the volume name TBA2).

5. Remove the formatted disk and insert the *Third Book of Apple* disk into the drive. Be sure that the side labeled PRODOS is facing upward. Type L for LIST PRODOS DIRECTORY. For the DIRECTORY PATHNAME prompt, type /AAP3BK and press Return. When the directory appears, write down all the filenames you see and then press Return. Now write down any additional filenames you see and press Esc.
6. Type C for COPY FILES. For the COPY PATHNAME prompt, type /AAP3BK/MENU, then press Return. For the TO PATHNAME prompt, type /TBA1/MENU, then press Return. Press Return again, and the file TUGAWAR will be placed into memory. When prompted, remove the *Third Book of Apple* disk and reinsert the TBA1 disk, then press Return to start the copy. When you see the message COPY COMPLETE, remove the newly formatted disk and reinsert the *Third Book of Apple* disk and go to the next filename you wrote down earlier. Repeat this procedure for all the files you wrote down, being careful not to mix up the disks and to keep the ProDOS side of the *Third Book of Apple* disk facing upward. To avoid typing the pathname and filename on each file, simply move the cursor over to the end of the prompt and type in the filenames only.
7. When you see the message

FILE NAME TOO LARGE

it's time to switch disks. Remove the TBA1 disk and replace it with the TBA2 disk. Continue with your copying, substituting TBA2 for TBA1 in the pathname. Be sure to copy MENU, AASET, and HROUT to both disks.

8. To use either disk, type RUN MENU at the] prompt and press Return.

Apple IIc ProDOS Users

1. Insert your *System Utilities* disk and turn on the computer.
2. From the System Utilities menu, type 6, then press Return, to select the Format a Disk option.
3. Press Return to select the built-in drive, then Return again to select ProDOS format.
4. Remove the *System Utilities* disk and insert a blank, unformatted disk in the drive.
5. Type in a name for your new disk volume, followed by Return.
6. When formatting is complete, press Esc to return to the main menu. Format another disk in the same manner, using a different name for the second disk volume. (You'll need two disks of your own to hold all the files on the *Third Book of Apple* disk.)
7. Remove the newly formatted disk and insert your *System Utilities* disk in the drive (it will be the source disk).
8. Type 1, then Return, to select the Copy Files option. Press Return twice more to specify the built-in drive for both the source and destination disks.
9. Press Return to copy some of the files on the disk.
10. Move the cursor up or down until the < > indicator is on the file called PRODOS.
11. Mark this file for copying by pressing the cursor-right key.
12. Mark the file BASIC.SYSTEM for copying by using the method explained in steps 10 and 11.
13. Press Return. The PRODOS file will be copied into memory.
14. When prompted, remove the *System Utilities* disk and insert the first newly formatted disk in the drive (it is the destination disk). Part of the PRODOS file will now be copied to your disk.
15. When prompted, remove your disk and again insert the *System Utilities* disk, then press Return. The remainder of the PRODOS file will be copied into memory.
16. When prompted, remove the *System Utilities* disk, insert your newly formatted disk, and press Return.
17. Repeat steps 15 and 16 to copy the BASIC.SYSTEM file to your disk.
18. Press Esc to return to the System Utilities main menu. Follow steps 7-17 to place the files PRODOS and BASIC.SYSTEM onto your second formatted disk.

19. Remove your disk and insert the *Third Book of Apple* disk into the drive. Be sure you have the side of the disk labeled ProDOS facing upward.
20. Type 1, then Return, to select the Copy Files option. Then press Return twice more to specify the built-in drive for both the source and destination disks. Select <ALL> by moving the cursor right, then pressing Return.
21. You will be prompted to switch back and forth between the *Third Book of Apple* disk (the source disk) and your disk containing the PRODOS and BASIC.SYSTEM files. Numerous swaps will be required to complete the copying, so be careful not to get the disks mixed up. Also, be sure that you don't turn over the *Third Book of Apple* disk; always keep the side with the ProDOS label facing upward when you insert it into the drive.
22. When you see the message which begins

Disk Is Full

- it's time to switch the destination disk. From this point, use the TBA2 disk. Continue with your copying.
23. You'll need to copy MENU, AASET, and HROUT to TBA2. Press Return to return to Copy Files. Press Return twice more to specify the built-in drive. Select <SOME> and press Return. Use the arrow keys to select the files MENU, AASET, and HROUT. Press Return and follow the screen prompts.
 24. To use either disk, type RUN MENU at the] prompt and press Return.

Index

- "Advanced Mousification" program 264-71
- Alquerque board game 33
- ampersand (&) 297
- animation 201-5
- "Animator2" graphics file program 205-9
- "Apple Animator" program v, 201-15
 - source code 209-15
- "Apple Automatic Proofreader" program 321-24
- "Apple Disk Booster" program 292-95
- "Apple Disk Duper" program 313-14
- Apple hi-res screen dump 222-26
- "Apple Keyboard Customizer" program 296-301
- "Apple MLX" program 18, 40, 173, 325-30
- Apple screen editing 286-89
- "Applesoft List Enhancer" program 315-16
- "Apple SpeedCalc" program 77-131
 - command summary 94
 - cursor movement 79-80
 - data formats 88-89
 - DOS 3.3 version 78, 94-113
 - editing 86-87
 - formula data 84-86
 - functions 85-86
 - keyboard commands 80-82
 - macro editing 89-90
 - memory available 90-91
 - numeric data 82-83
 - preparing 78-79
 - printing 92
 - ProDOS version 78, 114-32
 - recalculation 87-88
 - text data 84
- "Apple II Mouse Demonstration" program 259-64
- "Apple II Pull-Down Menus" program 246-49
- applications 77-197
- "ApWriter" program 144-52
 - command summary 149
 - margins 147
 - printer compatibility 148
 - tabs 146-47
- ASCII code 297
- astronomy 57-61
- auditory feedback, keyboard 283-85
- "BASIC Line Editor" program 286-91
 - commands 287-88
- BASIC memory, relocating 18, 78
- BRUN command 18, 41, 174
- Caps Lock key 325
- cell, spreadsheet 79, 80
- changing entries, "Apple SpeedCalc" and 83
- checksum 322
- CHR\$(4) function 144
- command summary, "Apple SpeedCalc" 94
- command summary, "ApWriter" 149
- COMPUTE!'s *Third Book of Apple* disk 331-39
- Ctrl (control) key 322
- cursor movement, "Apple SpeedCalc" 79-80
- cursor movement, "Dr. Disk" 175
- customized keyboard, saving through reboot 299
- data formats, "Apple SpeedCalc" 88-89
- DATA statement 11, 320
- Dazzle Draw program 216
- disk
 - backing up 313
 - COMPUTE!'s *Third Book of Apple* 331-39
 - directory (DOS 3.3) 178, 180-83
 - double-DOS 185-88
 - editing 173-88
 - operations, "Apple SpeedCalc" 91-92
- display, 40/80 column 145
- display modes 145
- "Dr. Disk" program 173-98
- DOS 3.3 90, 173, 284, 292, 319, 332-34
 - and ProDOS, on one disk 185-88
 - commands, "Dr. Disk" 174-76
- double-DOS disk 185-88
 - cursor movement 175
 - discovering forgotten password with 177-79
 - DOS 3.3 source code 188-92
 - printer slot and 174
 - ProDOS source code 193-97
 - undeleting files with 179-85
- "DUMP Example" program 226
- "DUMP" program 222-26
- editing, "Apple SpeedCalc" 86-87
- education 53-73
- escape mode editing 275-79, 286
- extra tracks, formatting on disk 292-94
- Fantavision program 216
- Filer ProDOS program 184
- file structure, ProDOS 184-85

formula data, "Apple *SpeedCalc*" 84-86
 functions, "Apple *SpeedCalc*" 85-86
 graphics 201-25
 Halley's Comet 60-61
 "Help Screen Editor" program 302-7
 help screens 302-4
 hexadecimal notation 175
 "Hickory, Dickory, Dock" program 53-56
 high-resolution graphics page 311
 "High Rise" program 18-32
 "Hi-Res Graphics Aid" program 216-21
 HTAB function 144
 INPUT statement 251
 joystick 40, 253-54, 258
 "Joystick Modifications" program 274
 keyboard, customized, saving through reboot 299
 keyboard, customizing *v*, 296-99
 keyboard commands, "Apple *SpeedCalc*" 80-82
 keyboard reference (chart), "Apple *SpeedCalc*" 81
 "Keynote" program 283-85
 Macintosh computer *v*, 229, 241
 macro editing, "Apple *SpeedCalc*" and 89-90
 margins, "ApWriter" 147
 mathematical operators, "Apple *SpeedCalc*" and 85
 "Memo Diary" program 133-43
 running first time 133-34
 memory available with "Apple *SpeedCalc*" 90-91
 memory mapping 251
 memory partitioning 308-11
 menus, pull-down *v*, 241-44
 "Miami Ice" program *v*, 39-50
 mouse 243
 adding to Applesoft programs 250-59
 editing functions 256-58
 interface card 250
 sensitivity 253
 Mousetext special graphics characters 145
 "MOUSEY File" program 271-73
 "Mousor" program 275-79
 "MOVE Routine" program 244-46
 "MultiMemory" program *v*, 308-12
 numeric data, "Apple *SpeedCalc*" and 82-83
 Open Apple key 80-82
 emulating on Apple II and Apple II+ 82
 paddle 40
 partitions 308-11
 password, forgotten, discovering with "Dr. Disk" 177-79
 pattern recognition 70-71
 personal finance 153-61
 pivot date, "Memo Diary" and 135-36
 PR#1 command 144
 printer
 activating 144
 compatibility, "ApWriter" and 148
 slot, "Dr. Disk" and 174
 printing, "Apple *SpeedCalc*" 92
 PRINT statement 144
 ProDOS 91, 173, 284, 313, 319, 334-39
 and DOS 3.3, on one disk 185-88
 commands, "Dr. Disk" 176-77
 file structure, ProDOS 184-85
 program listings, enhancing 315
 pull-down menus *v*, 241-44
 "Puzzler" program 70-74
 modifying 71
 recalculation, "Apple *SpeedCalc*" and 87-88
 recreation 3-49
 ROM editor 286
 screen, "Apple *SpeedCalc*" 79
 screen editing 275-79
 "Screen Editor Loader" program 305
 sector, disk 175
 "Skyscape" program *v*, 57-69
 "SpeedScript File Converter" program 77, 93
 "SpeedScript" word processor 77, 93
 spreadsheet, electronic *v*, 77-93
 "Switchbox" program 7-17
 tabs, "ApWriter" 146-47
 text data, "Apple *SpeedCalc*" 84
 "TEXT Filemaker" program 274
 time, telling 53
 track 175
 "Tug-A-War" program 3-6
 typing in programs 319-20
 undeleting a file under DOS 3.3, "Dr. Disk" and 179-83
 undeleting a file under ProDOS, "Dr. Disk" and 183-85
 uppercase 319
 Volume Bit Map 184, 187
 Volume Table of Contents. *See* VTOC
 VTAB function 144
 VTOC 179-83, 187, 292-93
 wedge 285
 windows *v*, 229-40
 "Windows in BASIC" program 238-40
 "Windows" program 229-40
 "Witching Hour, The" program 33-38
 word processors, uses and limitations of 144
 "Your Personal Ledger" program 153-72

To order your copy of *COMPUTE!'s Third Book of Apple Disk*, call our toll-free US order line: 1-800-346-6767 (in NY 212-887-8525) or send your prepaid order to:

COMPUTE!'s Third Book of Apple Disk
COMPUTE! Publications
P.O. Box 5038
F.D.R. Station
New York, NY 10150

All orders must be prepaid (check, charge, or money order). NC residents add 4.5% sales tax.

Send _____ copies of *COMPUTE!'s Third Book of Apple Disk* at \$12.95 per copy.

Subtotal \$_____

Shipping and Handling: \$2.00/disk \$_____

Sales tax (if applicable) \$_____

Total payment enclosed \$_____

☐ Payment enclosed

☐ Charge ☐ Visa ☐ MasterCard ☐ American Express

Acct. No. _____ Exp. Date _____
(Required)

Name _____

Address _____

City _____ State _____ Zip _____

Please allow 4-5 weeks for delivery.

To order your copy of CONQUEST, West Book or Paper Back,
call our hotline 1-800-426-6747 on the 212-
527-5122 or send your 1-400-512-5122

CONQUEST, P.O. Box 5122
CONQUEST Publications
P.O. Box 5122

New York, NY 10122

Attention: Please send me a copy of CONQUEST, P.O. Box 5122,
New York, NY 10122

Name _____
Address _____

City _____
State _____
Zip _____
Country _____

Payment enclosed _____
Charge to my MasterCard or American Express _____

Card No. _____
Exp. Date _____

Name _____

Address _____

City _____

Please send me a copy of _____

COMPUTE! Books

Ask your retailer for these **COMPUTE! Books** or order directly from **COMPUTE!**.

Call toll free (in US) **1-800-346-6767** (in NY 212-887-8525) or write COMPUTE! Books, P.O. Box 5038, F.D.R. Station, New York, NY 10150.

Quantity	Title	Price*	Total
_____	Becoming a MacArtist (80-9)	\$17.95	_____
_____	COMPUTE!'s Apple Games for Kids (91-4)	\$12.95	_____
_____	COMPUTE!'s First Book of Apple (69-8)	\$12.95	_____
_____	COMPUTE!'s Guide to Telecomputing on the Apple (98-1)	\$ 9.95	_____
_____	COMPUTE!'s Kids and the Apple (76-0)	\$12.95	_____
_____	Easy BASIC Programs for the Apple (88-4)	\$14.95	_____
_____	MacTalk: Telecomputing on the Macintosh (85-X)	\$14.95	_____
_____	SpeedScript: The Word Processor for Apple Personal Computers (000)	\$ 9.95	_____
_____	The Apple IIc: Your First Computer (001)	\$ 9.95	_____
_____	Apple Machine Language for Beginners (002)	\$14.95	_____
_____	COMPUTE!'s Second Book of Apple (008)	\$12.95	_____
_____	MacOffice: Using the Macintosh for Everything (006)	\$14.95	_____
_____	MacIdeas (015-7)	\$14.95	_____
_____	Using Your Macintosh: Beginning Microsoft BASIC and Applications (021-1)	\$16.95	_____
_____	Apple II Applications: 40 Programs for Your Apple (016-5)	\$14.95	_____
_____	Advanced Macintosh BASIC Programming (030-0)	\$16.95	_____
_____	COMPUTE!'s Third Book of Apple (063-7)	\$14.95	_____

*Add \$2.00 per book for shipping and handling.
Outside US add \$5.00 air mail or \$2.00 surface mail.

NC residents add 4.5% sales tax. _____
Shipping & handling: \$2.00/book _____
Total payment _____

All orders must be prepaid (check, charge, or money order).
All payments must be in US funds.

☐ Payment enclosed.

Charge ☐ Visa ☐ MasterCard ☐ American Express

Acct. No. _____ Exp. Date _____
(Required)

Name _____

Address _____

City _____ State _____ Zip _____

*Allow 4-5 weeks for delivery.
Prices and availability subject to change.
Current catalog available upon request.

COMPUTE BOOKS

For your interest in these COMPUTE BOOKS or orders

please write to COMPUTE

1000 North 1st Street, Suite 100, St. Paul, MN 55101

or call (612) 222-1000, ext. 100 or 101

or write to COMPUTE BOOKS, Inc.

1000 North 1st Street, Suite 100, St. Paul, MN 55101

or call (612) 222-1000, ext. 100 or 101

or write to COMPUTE BOOKS, Inc.

1000 North 1st Street, Suite 100, St. Paul, MN 55101

or call (612) 222-1000, ext. 100 or 101

or write to COMPUTE BOOKS, Inc.

1000 North 1st Street, Suite 100, St. Paul, MN 55101

or call (612) 222-1000, ext. 100 or 101

or write to COMPUTE BOOKS, Inc.

1000 North 1st Street, Suite 100, St. Paul, MN 55101

or call (612) 222-1000, ext. 100 or 101

or write to COMPUTE BOOKS, Inc.

1000 North 1st Street, Suite 100, St. Paul, MN 55101

or call (612) 222-1000, ext. 100 or 101

or write to COMPUTE BOOKS, Inc.

1000 North 1st Street, Suite 100, St. Paul, MN 55101

or call (612) 222-1000, ext. 100 or 101

or write to COMPUTE BOOKS, Inc.

1000 North 1st Street, Suite 100, St. Paul, MN 55101

or call (612) 222-1000, ext. 100 or 101

or write to COMPUTE BOOKS, Inc.

1000 North 1st Street, Suite 100, St. Paul, MN 55101

or call (612) 222-1000, ext. 100 or 101

or write to COMPUTE BOOKS, Inc.

1000 North 1st Street, Suite 100, St. Paul, MN 55101

or call (612) 222-1000, ext. 100 or 101

or write to COMPUTE BOOKS, Inc.

1000 North 1st Street, Suite 100, St. Paul, MN 55101

or call (612) 222-1000, ext. 100 or 101

or write to COMPUTE BOOKS, Inc.

1000 North 1st Street, Suite 100, St. Paul, MN 55101

or call (612) 222-1000, ext. 100 or 101

or write to COMPUTE BOOKS, Inc.

1000 North 1st Street, Suite 100, St. Paul, MN 55101

or call (612) 222-1000, ext. 100 or 101

or write to COMPUTE BOOKS, Inc.

1000 North 1st Street, Suite 100, St. Paul, MN 55101

or call (612) 222-1000, ext. 100 or 101

or write to COMPUTE BOOKS, Inc.



An Extraordinary Collection

Some of the best Apple software you'll find anywhere comes from COMPUTE! Publications.

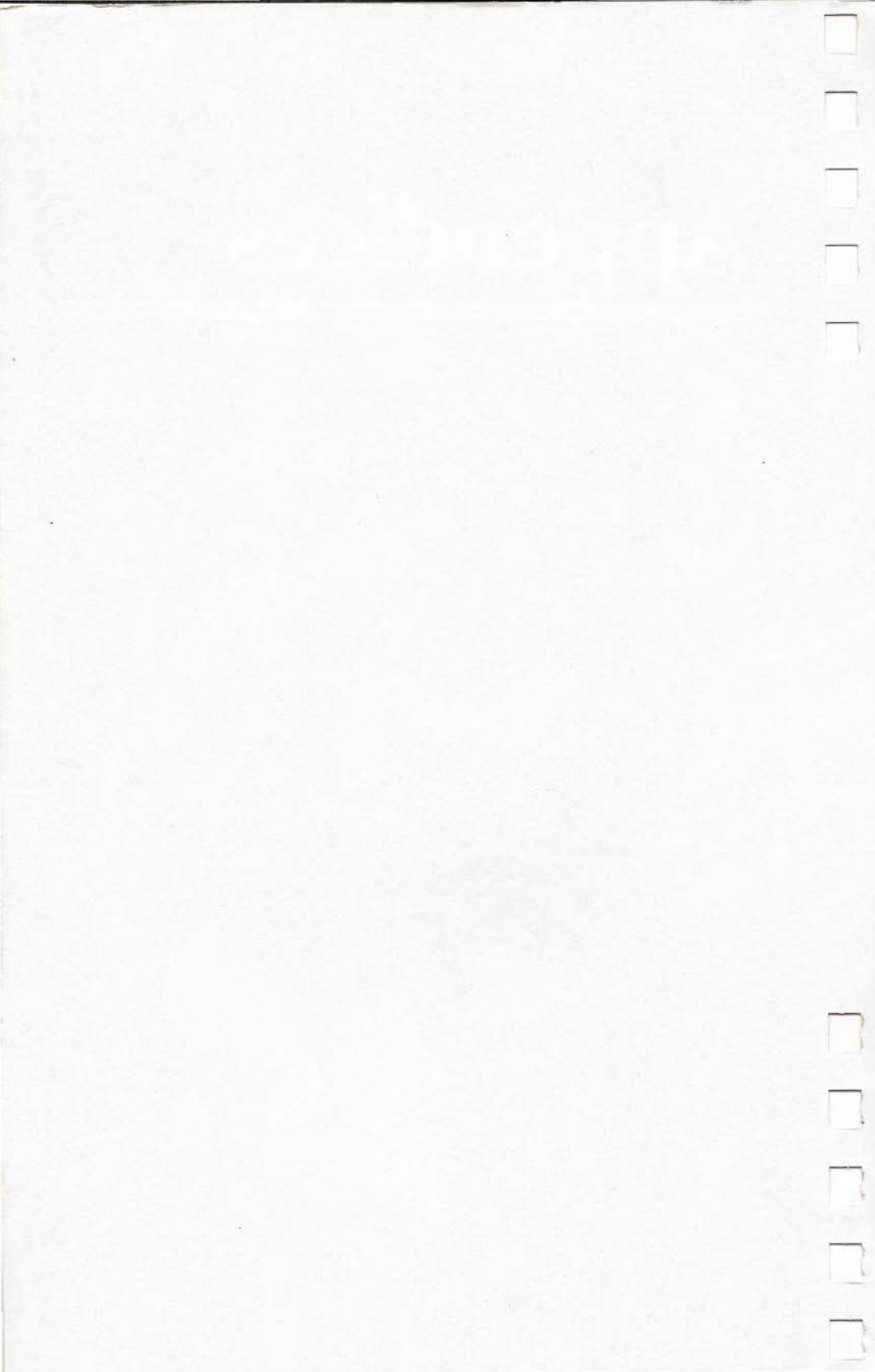
High-quality, ready-to-type-in programs from *COMPUTE!* magazine and *COMPUTE!'s Apple Applications Special* have been chosen for our third Apple anthology, *COMPUTE!'s Third Book of Apple*. Inside you'll find exciting arcade games, impressive home and business applications, colorful graphics creators, useful programming utilities, and entertaining educational software.

- *SpeedCalc*, an all machine language spreadsheet that rivals commercial software in ease of use, power, and features.
- "High Rise," an arcade game of running, dodging, and falling. Perhaps our best Apple game yet.
- "Hickory, Dickory, Dock" helps youngsters tell time with a colorful clock and hands that move.
- "Apple Animator" puts you in charge of your own animation studio.
- Three programs magically make your Apple II act like a Macintosh, adding pull-down menus, windows, and mouse movement.
- And programming utilities and tutorials that customize the keyboard, simplify editing, and duplicate disks at high speed.

Understandable instructions, thoroughly tested programs, and error-checking aids that simplify typing in our Apple listings make *COMPUTE!'s Third Book of Apple* the best software bargain around.

All the programs in this book are available on a companion disk. See the coupon in the back for details.

Appendices



Guide to Typing In Programs

What Is a Program?

A computer cannot perform any task by itself. Like a car without gas, a computer has *potential*, but without a program, it isn't going anywhere. Most of the programs in this book are written in a computer language called Applesoft BASIC. It's easy to learn and works on the Apple II, II+, IIe, and IIfx.

BASIC Programs

Computers can be picky. Unlike the English language, which is full of ambiguities, BASIC usually has only one right way of stating something. Every letter, character, and number is significant. A common mistake is substituting a letter such as *O* for the numeral 0, a lowercase *l* for the numeral 1, or an uppercase *B* for the numeral 8. Also, you must enter all punctuation marks such as colons and commas just as they appear in the listings. Spacing can be important. To be safe, type in the programs exactly as they appear. Unlike other program listings you may have seen, those in this book have no special characters which you need to interpret. Simply enter the programs as they appear.

DOS 3.3 and ProDOS

Unless otherwise mentioned in the program's accompanying article, it doesn't matter whether you have DOS 3.3 or ProDOS. You can enter the programs with either DOS active in your Apple. Of course, you can run a typed-in program only with the DOS system it was entered with.

Uppercase

You'll notice that all the program listings are entirely in uppercase. If you have an Apple IIe or IIfx, however, which allows both uppercase and lowercase, you can change text which appears in PRINT statements if you want. A program such as "Your Personal Ledger," for instance, could be modified so that the screen displays appear in both uppercase and lowercase.

DATA Statements

Some programs contain one or more sections of DATA statements. These lines provide information needed by the program. Some DATA statements contain actual programs (called machine language); others contain graphics codes. These lines are especially sensitive to errors.

If a single number in any one DATA statement is mistyped, your machine could lock up, or crash. The keyboard may seem dead, and the screen may go blank. Don't panic—no damage is done. To regain control, you have to turn off your computer, then turn it back on. This will erase whatever program was in memory, *so always save a copy of your program before you run it*. If your computer crashes, you can load the program and look for your mistake.

Sometimes, a mistyped DATA statement will cause an error message when the program is run. The error message may refer to the program line that READs the data. *The error is still in the DATA statements, though.*

Get to Know Your Machine

You should familiarize yourself with your computer before attempting to type in a program. Learn the statements you use to store and retrieve programs from tape or disk. You'll want to save a copy of your program so that you won't have to type it in every time you use it. Learn to use your machine's editing functions. How do you change a line if you make a mistake? You can always retype the line, but you at least need to know how to use the left- and right-arrow keys. It's all explained in your computer's manuals.

A Quick Review

1. Type in the program, a line at a time, in order. Press Return at the end of each line. Use the left arrow to correct mistakes.
2. Check the line you've typed against the line in the book. You can check the entire program again if you get an error when you run the program.

Apple Automatic Proofreader

Tim Victor

It's easier than ever to enjoy programs for Apple II series computers. "Apple Automatic Proofreader," an error-checking program for the Apple II, II+, IIe, and IIC with either DOS 3.3 or ProDOS, alerts you to almost every typing mistake you might make.

"Apple Automatic Proofreader" will help you type in program listings without typing mistakes. It's a short error-checking program that hides itself in memory and attaches to your Apple's operating system. Each time you press Return to enter a program line, this routine displays a two-digit checksum at the top of your screen. If you've typed the line correctly, the checksum on your screen matches the one in the printed listing—it's that simple. You don't have to use the Proofreader to enter listings, but doing so greatly reduces the chance of making a typo.

Getting Started

First, type in the Apple Automatic Proofreader program following this article. The Proofreader can't check itself before it's done, so you'll have to be extra careful to avoid mistakes.

The Proofreader checks which operating system you're running before it hooks up the checksum routine, so you can type it in with either DOS 3.3 or ProDOS. If you want to use the Proofreader with both operating systems, you won't have to retype it. All you need is a utility to copy a file between disks with different formats, such as the one provided on the ProDOS *System Utilities* disk.

As soon as you finish typing the Proofreader, save at least two copies. This is very important, because the Proofreader erases the BASIC portion of itself when you run it, leaving only the machine language portion in memory.

Now, type RUN and hit Return. The Proofreader clears the screen, loads the machine language routine, displays the message PROOFREADER ACTIVATED, erases the BASIC portion of itself, and ends. If you type LIST and press Return,

you'll see that no BASIC program is in memory. The computer is ready for you to type in a new BASIC program.

Entering Programs

Once the Proofreader is activated, you can begin typing in a BASIC program as usual. Every time you finish typing a line and press Return, the Proofreader displays a two-digit checksum number in the upper-left corner of the screen. Compare this checksum with the two-digit checksum printed next to the corresponding line in the program listing. If the numbers match, you can be pretty certain the line was typed correctly. Otherwise, check for your mistake and type the line again.

A common mistake when entering BASIC programs on the Apple occurs when you accidentally press a key while holding down the Ctrl (Control) key. This adds an invisible control character to the line you are typing. If you don't find it before you run the program, this stray character may cause a SYNTAX ERROR or other mysterious behavior. Fortunately, the Proofreader detects the presence of these invisible control characters, displaying a checksum that doesn't match the one in the listing. So it's always a good idea to retype a line if the checksums don't match, even though you might not see any difference in the lines themselves.

The Proofreader ignores space characters, so you can omit spaces between keywords and still see a matching checksum. Spaces are important only between the quotation marks of PRINT statements or string assignments. If you accidentally type too many spaces or leave some out, this is the only mistake the Proofreader won't catch. For this reason, you should be extra careful when entering text within quotation marks.

Before running another BASIC program, it's a good idea to turn off the Proofreader by holding down the Ctrl key while pressing the Reset button. The machine language part of the Proofreader is kept in memory starting at address 768 (\$300 hexadecimal). This location is out of BASIC's way, but a lot of other programs use this same place for their machine language subroutines. Disable the Proofreader to avoid conflicts.

How It Works

When the Applesoft BASIC interpreter needs to get a line of

input from the keyboard, it calls a machine language routine in the Apple's read only memory (ROM) called GETLN. GETLN, in turn, calls the operating system to get a single keypress, which it stores in an input buffer. If the Return key was pressed, GETLN ends, leaving one new line for the BASIC interpreter in the input buffer. Otherwise, it repeats the process, asking for another keypress.

The operating system normally gets individual keystrokes from a ROM routine called KEYIN, but the Proofreader changes this. When the Proofreader is installed, the operating system calls the checksum routine instead, and the checksum routine asks KEYIN for a character. If any key other than Return was pressed, the checksum routine just passes it on to the operating system, which gives it to GETLN. But if Return *was* pressed, the checksum routine examines the contents of GETLN's input buffer, which now contains an entire line of input, to calculate the checksum that it displays at the top of the screen.

One very common typing mistake is transposition—typing two successive characters in the wrong order, like *PIRNT* instead of *PRINT*. A checksum program that merely adds the codes of the characters in a line can detect only the presence or absence of a character, not transposition errors. Because the Apple Proofreader uses a sophisticated formula to compute checksums, it alerts you to transposed keystrokes.

The Apple Automatic Proofreader detects almost every possible typing mistake, including transpositions, missing or extra characters, accidental control characters, and incorrect line numbers. Typing *COMPUTE!* Publications' programs into your Apple computer has never been easier.

Apple Automatic Proofreader

```

52 100 C = 0: FOR I = 768 TO 768 + 68: READ A: C = C +
      A: POKE I, A: NEXT
80 20 IF C < > 7258 THEN PRINT "ERROR IN PROOFREADER
      DATA STATEMENTS": END
00 30 IF PEEK (190 * 256) < > 76 THEN POKE 56, 0: POK
      E 57, 3: CALL 1002: GOTO 50
70 40 PRINT CHR$ (4); "IN#A$300"
24 50 POKE 34, 0: HOME : POKE 34, 1: VTAB 2: PRINT "PR
      OOFREADER INSTALLED"
FE 60 NEW
52 100 DATA 216, 32, 27, 253, 201, 141

```



```
10 110 DATA 208,60,138,72,169,0
75 120 DATA 72,189,255,1,201,160
FA 130 DATA 240,8,104,10,125,255
47 140 DATA 1,105,0,72,202,208
1B 150 DATA 238,104,170,41,15,9
AF 160 DATA 48,201,58,144,2,233
08 170 DATA 57,141,1,4,138,74
9E 180 DATA 74,74,74,41,15,9
B5 190 DATA 48,201,58,144,2,233
E3 200 DATA 57,141,0,4,104,170
A9 210 DATA 169,141,96
```

Apple MLX Machine Language Entry Program

Tim Victor

Machine language programs are difficult to enter into your computer. To make this chore easier and to eliminate typing mistakes, COMPUTE! Publications presents a machine language entry program for the Apple II, II+, IIe, and IIfx computers, using either the DOS 3.3 or ProDOS operating system.

A machine language program is usually listed as a long series of numbers. It's hard to keep your place and even harder to avoid making mistakes as you type in the listing since an incorrect line looks almost the same as a correct one. To reduce the problems associated with typing in machine language programs, we've presented them as MLX listings which can be entered using the "Apple MLX" editor.

MLX checks your typing on a line-by-line basis. It won't let you enter inappropriate characters, and it won't let you continue if there's a mistake in a line or even if you're trying to enter a line or digit out of sequence. You don't have to know anything about machine language to use it. In other words, MLX makes machine language program entry almost foolproof.

Using MLX

Type in and save MLX to disk (you'll want to use it to enter a number of programs in this book as well as some of those listed in *COMPUTE!* magazine and *COMPUTE!'s Apple Applications Special* issues). It doesn't matter whether you type it in on a disk formatted for DOS 3.3 or ProDOS. Programs entered with MLX, however, must be saved to a disk formatted with the same operating system as MLX itself.

If you have an Apple IIe or IIfx, make sure that the key marked Caps Lock is in the down position. Type RUN. You'll be asked for the starting and ending addresses of the machine language program. These values are given at the beginning of

the machine language program listing and in the program's accompanying article. Find them and type them in.

The next thing you'll see is a menu asking you to select a function. The first is (E)NTER DATA. If you're just starting to type in a program, pick this. Press the E key, and the program asks for the address where you want to begin entering data. Type the first number in the first line of the program listing if you're just starting or the line number where you left off if you've already typed in part of a program. Hit the Return key and begin entering the data.

Once you're in enter mode, MLX will print the address for each program line for you. You then type in all nine numbers on that line, beginning with the first two-digit number after the colon (:). Each line represents eight bytes and a checksum. When you enter a line and hit Return, MLX recalculates the checksum from the eight bytes and the address. If you enter more than or fewer than nine numbers, or if the checksum doesn't exactly match, MLX erases the line you just entered and prompts you again for the same line.

Invalid Characters Banned

MLX is fairly flexible about how you type in the numbers. You can put extra spaces between numbers or leave the spaces out entirely, compressing a line into 18 keypresses. Be careful not to put a space between two digits in the middle of a number. MLX will read two single-digit numbers instead of one two-digit number (F 6 means F and 6, not F6).

You can't enter an inappropriate character with MLX. Only the numerals 0-9 and the letters A-F can be typed in. If you press any other key (with some exceptions noted below), nothing happens. This safeguards against entering extraneous characters. Even better, MLX checks for transposed characters. If you're supposed to type in A0 and instead enter 0A, MLX will catch your mistake.

MLX also checks to make sure you're typing in the right line. The address (the number to the left of the colon) is part of the checksum recalculation. If you accidentally skip a line and try to enter incorrect values, MLX won't let you continue. Just make sure you enter the correct starting address; if you don't, you won't be able to enter any of the following lines. MLX will stop you.

Editing Features

MLX also includes some editing features. The left- and right-arrow keys allow you to back up and go forward on the line you're entering so that you can retype data. Pressing the Ctrl (Control) key and the D key at the same time (*Delete*) removes the character under the cursor, shortening the line by one character. Pressing the Ctrl key and the I key simultaneously (*Insert*) puts a space under the cursor and shifts the rest of the line to the right, making the line one character longer. If the cursor is at the right end of the line, neither Ctrl-D nor Ctrl-I has any effect.

When you've entered the entire listing (up to the ending address that you specified earlier), MLX automatically leaves enter mode and redisplay the functions menu. If you want to leave enter mode before then, press the Return key when MLX prompts you with the address of a new line.

Display Data

The second menu choice, (D)ISPLAY DATA, examines memory and shows the contents in the same format as the program listing. You can use it to check your work or to see how far you've got. When you press the D key, MLX asks you for a starting address. Type in the address of the first line that you want to see and hit Return. MLX displays program lines until you press any key or until it reaches the end of the program.

Save and Load

Two menu selections are provided to let you save programs to disk and load them back into the computer. These are (S)AVE FILE and (L)OAD FILE. MLX asks you for the name of the file which contains the program. The first time you save a machine language program, there won't be a file on the disk containing the program. Whatever name you type in will be the name of a new file that's created.

The message DISK ERROR appears during a save or load if a problem is detected. If you're not sure why a disk error has occurred, check the disk drive. Make sure there's a formatted disk in the drive and that it was formatted by the same operating system that you're using for MLX (ProDOS or DOS 3.3). If you're trying to save a file and see an error message, the disk might be full. Either save the file on another disk or quit MLX (by pressing Q), delete an old file or two, then run

MLX again. Your typing should still be safe in memory. If the error message appears during a load, you may have specified a filename that doesn't exist on the disk.

Quit

The Quit menu option has the obvious effect—it stops MLX and enters BASIC. (Of course, you can also press Ctrl-Reset to get out of MLX.)

The Finished Product

When you've finished typing all the data for a machine language program and saved your work, you're ready to see the results. The instructions for loading and using the finished product vary from program to program. You'll almost always load and run an MLX-generated program by typing `BRUN filename` (or sometimes just `BLOAD`).

An Ounce of Prevention

By the time you finish typing in the data for a long program, you may have several hours invested in the project. Don't take chances—use the "Apple Automatic Proofreader" to enter MLX, and then test your copy *thoroughly* before first using it to enter any significant amount of data. Make sure all the menu options work as they should. Enter fragments of the program starting at several different addresses, then use the Display option to verify that the data has been entered correctly. And be sure to test the Save and Load options several times to insure that you can recall your work from disk. Don't let a simple typing error in MLX cost you several nights of hard work.

Line 100 of MLX traps all errors to line 610. If MLX is typed in correctly, then only disk errors should be encountered. A disk error message when you're not trying to access the drive—for example, when you first start entering data—indicates a typing error in the MLX program itself. If this occurs, hit Ctrl-Reset to break out of MLX and carefully compare your entry against the printed listing.

Apple MLX: Machine Language Entry Program

For mistake-proof program entry, use the "Apple Automatic Proofreader" (Appendix B) to type in this program.

```
## 100 N = 9: HOME : NORMAL : PRINT "APPLE MLX": POK
E 34,2: ONEKR GOTO 610
```



```

CC 110 VTAB 1: HTAB 20: PRINT "START ADDRESS";: GOSU
B 530: IF A = 0 THEN PRINT CHR$ (7): GOTO 110
BC 120 S = A
EC 130 VTAB 2: HTAB 20: PRINT "END ADDRESS ";: GOSU
B 530: IF S >= A OR A = 0 THEN PRINT CHR$ (7
): GOTO 130
20 140 E = A
B5 150 PRINT : PRINT "CHOOSE:(E)NTER DATA";: HTAB 22
: PRINT "(D)ISPLAY DATA": HTAB 8: PRINT "(L)O
AD FILE (S)AVE FILE (Q)UIT": PRINT
AE 160 GET A$: FOR I = 1 TO 5: IF A$ < > MID$ ("EDLS
Q",I,1) THEN NEXT : GOTO 160
F3 170 ON I GOTO 270,220,180,200: POKE 34,0: END
AF 180 INPUT "FILENAME: ";A$: IF A$ < > "" THEN PRIN
T CHR$ (4);"BLOAD";A$;"A";S
A1 190 GOTO 150
6D 200 INPUT "FILENAME: ";A$: IF A$ < > "" THEN PRIN
T CHR$ (4);"BSAVE";A$;"A";S;"L";E - S
92 210 GOTO 150
C2 220 GOSUB 590: IF B = 0 THEN 150
9E 230 FOR B = B TO E STEP 8:L = 4:A = B: GOSUB 580:
PRINT A$;" ":L = 2
B5 240 FOR F = 0 TO 7:V(F + 1) = PEEK (B + F): NEXT
: GOSUB 560:V(9) = C
F2 250 FOR F = 1 TO N:A = V(F): GOSUB 580: PRINT A$"
":NEXT : PRINT : IF PEEK (49152) < 128 THE
N NEXT
94 260 POKE 49168,0: GOTO 150
CC 270 GOSUB 590: IF B = 0 THEN 150
48 280 FOR B = B TO E STEP 8
A6 290 HTAB 1:A = B:L = 4: GOSUB 580: PRINT A$;" ":
: CALL 64668:A$ = "" :P = 0: GOSUB 330: IF L =
0 THEN 150
F9 300 GOSUB 470: IF F < > N THEN PRINT CHR$ (7): G
OTO 290
27 310 IF N = 9 THEN GOSUB 560: IF C < > V(9) THEN P
RINT CHR$ (7): GOTO 290
72 320 FOR F = 1 TO 8: POKE B + F - 1,V(F): NEXT : P
RINT : NEXT : GOTO 150
8E 330 IF LEN (A$) = 33 THEN A$ = 0$:P = 0: PRINT CH
R$ (7);
22 340 L = LEN (A$):O$ = A$:O = P:L$ = "": IF P > 0
THEN L$ = LEFT$ (A$,P)
E0 350 R$ = "": IF P < L - 1 THEN R$ = RIGHT$ (A$,L
- P - 1)
55 360 HTAB 7: PRINT L$: FLASH : IF P < L THEN PRIN
T MID$ (A$,P + 1,1): NORMAL : PRINT R$;
7B 370 PRINT " ": NORMAL
E6 380 K = PEEK (49152): IF K < 128 THEN 380
C1 390 POKE 49168,0:K = K - 128

```



```

53 400 IF K = 13 THEN HTAB 7: PRINT A$; " ";: RETURN
8A 410 IF K = 32 OR K > 47 AND K < 58 OR K > 64 AND
    K < 71 THEN A$ = L$ + CHR$ (K) + R$: P = P + 1
C1 420 IF K = 4 THEN A$ = L$ + R$
5F 430 IF K = 9 THEN A$ = L$ + " " + MID$ (A$, P + 1,
    1) + R$
0A 440 IF K = 3 THEN P = P - (P > 0)
93 450 IF K = 21 THEN P = P + (P < L)
9D 460 GOTO 330
37 470 F = 1: D = 0: FOR P = 1 TO LEN (A$): C$ = MID$
    (A$, P, 1): IF F > N AND C$ < > " " THEN RETURN
E9 480 IF C$ < > " " THEN GOSUB 520: V(F) = J + 16 *
    (D = 1) * V(F): D = D + 1
5F 490 IF D > 0 AND C$ = " " OR D = 2 THEN D = 0: F =
    F + 1
0B 500 NEXT : IF D = 0 THEN F = F - 1
17 510 RETURN
55 520 J = ASC (C$): J = J - 48 - 7 * (J > 64): RETUR
    N
AB 530 A = 0: INPUT A$: A$ = LEFT$ (A$, 4): IF LEN (A$
    ) = 0 THEN RETURN
6F 540 FOR P = 1 TO LEN (A$): C$ = MID$ (A$, P, 1): IF
    C$ < "0" OR C$ > "9" AND C$ < "A" OR C$ > "Z"
    THEN A = 0: RETURN
2D 550 GOSUB 520: A = A * 16 + J: NEXT : RETURN
28 560 C = INT (B / 256): C = B - 256 * C - 255 * (C
    > 127): C = C - 255 * (C > 255)
20 570 FOR F = 1 TO 8: C = C * 2 - 255 * (C > 127) +
    V(F): C = C - 255 * (C > 255): NEXT : RETURN
0A 580 I = FRE (0): A$ = "": FOR I = 1 TO L: T = INT (
    A / 16): A$ = MID$ ("0123456789ABCDEF", A - 16
    * T + 1, 1) + A$: A = T: NEXT : RETURN
1F 590 PRINT "FROM ADDRESS ";: GOSUB 530: IF S > A 0
    R E < A OR A = 0 THEN B = 0: RETURN
0D 600 B = S + 8 * INT ((A - S) / 8): RETURN
8A 610 PRINT "DISK ERROR": GOTO 150

```

Disk Instructions

Typing in a long BASIC or machine language program can be a time-consuming task. Even with sophisticated error-checking programs like "Apple Automatic Proofreader" and "Apple MLX," you still have to spend hours in front of the computer.

That's why we've made available for purchase a disk containing all the programs in this book. To order the *COMPUTE!'s Third Book of Apple* companion disk, use the coupon in the back of this book, or call toll-free 1-800-346-6767 (in NY 1-212-887-8525).

Preparing a Purchased Disk

If you've bought *COMPUTE!'s Third Book of Apple* disk and simply put it in your computer's disk drive and turn the computer on, you'll see this message:

COMPUTE!'S THIRD BOOK OF APPLE

(C) COPYRIGHT 1986 ALL RIGHTS RESERVED

**INSERT A DOS 3.3 DISK THEN
PRESS ANY KEY TO BOOT**

or

**INSERT A PRODOS DISK THEN
PRESS ANY KEY TO BOOT**

The disk you purchased doesn't contain the version of DOS (Disk Operating System) necessary to start the Apple. You have two options. You can boot the system each time with a disk that does contain DOS 3.3 or ProDOS (look for a DOS 3.3 disk called *System Master* or a ProDOS disk called *ProDOS User's Disk*—one or the other came with your computer), then insert the *Third Book of Apple* disk and type **RUN MENU**.

A better way is to copy programs from the *Third Book of Apple* disk to your own disk, one that has DOS 3.3 or ProDOS already on it. It's not hard.

Note: *You'll need two disks to hold all the files on the Third Book of Apple disk in addition to the DOS 3.3 or ProDOS system files. Splitting the files between two disks means that you must only select from the MENU those programs copied to the specific disk. If you select a program which wasn't copied to that disk, you'll receive an error message.*

DOS 3.3 Users with Two Disk Drives

1. Insert your *System Master* disk in drive 1 and turn on the computer.
2. Insert the *Third Book of Apple* disk in drive 2. Be sure that the side labeled DOS 3.3 is facing upward.
3. Type **LOAD MENU,D2**, then press the Return key.
4. Remove the *Third Book of Apple* disk from drive 2, and insert a blank, unformatted disk in drive 2.
5. Type **INIT HELLO,D2**, then press Return. The disk in the drive will be formatted and the DOS system files will be written to the disk. When the formatting is completed, remove the disk from drive 2, place another blank, unformatted disk in drive 2, and type **INIT HELLO,D2** again. (You'll need two disks of your own to hold all the files on the *Third Book of Apple* disk in addition to the DOS system files.)
6. When the cursor reappears, type **BRUN FID,D1** and press Return.
7. Remove your *System Master* disk from drive 1, and insert the *Third Book of Apple* disk. Again, be sure that the side labeled DOS 3.3 is facing upward.
8. From the File Developer menu prompt, press 2, then Return, to select the CATALOG option. For the SOURCE SLOT? prompt, type 6, then Return, and for the DRIVE? prompt, type 1 and then Return.
9. Write down all the filenames (except MENU) from the *Third Book of Apple* disk, then press any key to get back to the File Developer menu.
10. Type 1, then press Return, to select the COPY FILES option. For the SOURCE SLOT? prompt, type 6, then Return, and for the DRIVE? prompt, type 1, then Return. For the DESTINATION SLOT? prompt, type 6, then Return, and for the DRIVE? prompt, type 2, then Return.
11. Type in the name of the file to be transferred, followed by Return, and hit any key to proceed with the copy. Then hit any key to return to the menu.
12. Type 2, then Return, if you need to refresh your memory about filenames (select slot 6, drive 1). Then hit any key for the menu.

13. Otherwise, repeat steps 10–12 until all files are copied. Slot and drive designations will default to your specifications unless changed, so you will not have to enter these again for each file. Change to the second formatted disk when you receive this message:

DISK FULL

CANCELLED

PRESS ANY KEY TO CONTINUE

14. Copy files from the *Third Book of Apple* disk to your second disk as you did before.
15. For the MENU to work, the files AASET and HROUT must be copied to both disks.

DOS 3.3 Users with One Disk Drive

1. Insert your *System Master* disk and turn on the computer.
2. Remove the *System Master* disk and insert the *Third Book of Apple* disk. Be sure that the side labeled DOS 3.3 is facing upward.
3. Type **LOAD MENU**, then press Return.
4. Remove the *Third Book of Apple* disk, and insert a blank, unformatted disk in the drive.
5. Type **INIT HELLO**, then press Return. The blank disk will be formatted and DOS system files will be written to the disk. When the formatting is completed, remove the disk from the drive, place another blank, unformatted disk in the drive, and type **INIT HELLO**, again. (You'll need two disks of your own to hold all the files on the *Third Book of Apple* disk plus the DOS system files.)
6. When the cursor reappears, remove the second newly formatted disk and insert your *System Master* disk.
7. Type **BRUN FID**, then Return.
8. Remove the *System Master* disk and insert the *Third Book of Apple* disk in the drive. Again, be sure that the side labeled DOS 3.3 is facing upward.
9. At the File Developer menu prompt, type 2, then press Return, for the CATALOG option. At the SOURCE SLOT? prompt, type 6, then Return, and for the DRIVE? prompt, type 1, then Return.
10. Write down all the filenames (except MENU) from the *Third Book of Apple* disk, then press any key to get back to the File Developer menu.

11. Type 1, then press Return, to select the COPY FILES option. At the SOURCE SLOT? prompt, type 6, then Return, and for the DRIVE? prompt, type 1, then Return. For the DESTINATION SLOT? prompt, type 6, Return, and for the DRIVE? prompt, type 1, Return.
12. Type in the filename of the first file to be transferred and press Return; then press any key *twice* to proceed with the transfer.
13. Remove the *Third Book of Apple* disk, insert your newly formatted disk, and press any key to make the copy.
14. Remove your disk and insert the *Third Book of Apple* disk again; then press any key to return to the File Developer menu.
15. Type 2, then press Return, if you need to see the list of filenames again (specify slot 6, drive 1).
16. Repeat steps 11–15 until all files are copied. Slot and drive designations will default to your specifications unless changed, so you will not have to enter these again for each file. As you swap disks back and forth, be very careful that you always insert the *Third Book of Apple* disk with the side labeled DOS 3.3 facing upward.
17. Change to the second formatted disk when you receive this message:
**DISK FULL
CANCELLED
PRESS ANY KEY TO CONTINUE**
18. Copy files from the *Third Book of Apple* disk to your second disk as you did before.
19. For the MENU program to work, the files AASET and HROUT must be copied to both disks.

ProDOS Users with Two Disk Drives

1. Put your *ProDOS User's Disk* in drive 1, and a blank, unformatted disk in drive 2, and turn on your computer.
2. From the master menu, type F for PRODOS FILER (UTILITIES). From the Filer menu, type V for VOLUME COMMANDS. From the Volume Commands menu, type F for FORMAT A VOLUME. Type in 6 for SLOT, 2 for DRIVE, and TBA1 as your NEW VOLUME NAME. (You can use another volume name if you want.) After you press Return, the formatting will begin. When you see the message FORMAT COMPLETE, press Esc twice to return to the

Filer menu. Remove the disk from drive 2, put another blank, unformatted disk in drive 2, and follow the above instructions again. This time, type **TBA2** as your NEW VOLUME NAME. (You'll need two disks of your own to hold all the files on the *Third Book of Apple* disk.) When the second disk is formatted, replace it with the first formatted disk.

3. From the Filer menu, type **F** for FILE COMMANDS, then **C** for COPY FILES. For the COPY PATHNAME prompt, type **/USERS.DISK/PRODOS** and press Return; for the TO PATHNAME prompt, type **/TBA1/PRODOS** and press Return. When you press Return again, the file PRODOS will be copied to your newly formatted disk. When you see the message COPY COMPLETE, press Return and copy the file BASIC.SYSTEM by typing **/USERS.DISK/BASIC.SYSTEM** for the COPY PATHNAME prompt; press Return. Then type **/TBA1/BASIC.SYSTEM** for the TO PATHNAME prompt and press Return. You can simply use the cursor to change the filename at the end of each prompt. Press Return to begin the copy. When you see the message COPY COMPLETE, press Esc. Follow the above instructions to copy the files PRODOS and BASIC.SYSTEM to your second disk (the one with the volume name TBA2).
4. When prompted, type **L** for LIST PRODOS DIRECTORY. Remove the *ProDOS User's Disk* from drive 1, and insert the *Third Book of Apple* disk. Be sure the side labeled ProDOS is facing upward. Now for the DIRECTORY PATHNAME prompt, type **/APP3BK** and press Return. When the directory appears, write down all the filenames, then press Return, and write down any other filenames that appear. Press Esc to return to the menu.
5. When prompted, press **C** for COPY FILES. For the COPY PATHNAME prompt, type **/AAP3BK/MENU** and press Return; for the TO PATHNAME prompt, type **/TBA1/MENU** and press Return. Press Return again, and the copy will begin. Repeat this procedure for each filename you wrote down earlier. Remember, you can cursor over the pathname and change only the filenames rather than typing in both each time.
6. When you see the message
FILE NAME TOO LARGE

it's time to switch disks in drive 2. Remove the TBA1 disk and replace it with the TBA2 disk. Continue with your copying, substituting TBA2 for TBA1 in the pathname. Be sure to copy MENU, AASET, and HROUT to both disks.

7. To use either disk, type **RUN MENU** at the] prompt and press Return.

ProDOS Users with One Disk Drive

1. Put your *ProDOS User's Disk* in the drive and turn on the computer.
2. From the master menu, type **F** for PRODOS FILER (UTILITIES). From the Filer menu, type **V** for VOLUME COMMANDS. From the Volume Commands menu, type **F** for FORMAT A VOLUME. Next, type **6** and press Return for the SLOT prompt, then type **1** and press Return for the DRIVE prompt. For the NEW VOLUME NAME prompt, type **TBA1** and press Return. (You can use something else if you want.) After you press Return, the formatting will begin. When you see the message **FORMAT COMPLETE**, press **Esc** twice to return to the Filer menu. Remove the disk from drive 2, put another blank, unformatted disk in drive 2, and follow the above instructions again. This time, type **TBA2** as your NEW VOLUME NAME. (You'll need two disks of your own to hold all the files on the *Third Book of Apple* disk.)
3. When you see the message **FORMAT COMPLETE** for the second time, remove the newly formatted disk and reinsert the *ProDOS User's Disk* into the drive. Then, press **Esc** twice to return to the Filer menu.
4. Type **F** for FILE COMMANDS, then **C** for COPY FILES. For the COPY PATHNAME prompt, type **/USERS.DISK/PRODOS** and press Return. For the TO PATHNAME prompt, type **/TBA1/PRODOS** and press Return. Press Return again, and the file PRODOS will be placed into memory. When prompted, remove the *ProDOS User's Disk* and insert your first formatted disk, then press Return. A copy of the file PRODOS will be placed on your newly formatted disk. When you see the **COPY COMPLETE** message, reinsert the *ProDOS User's Disk* and copy the file **BASIC.SYSTEM** by typing **/USERS.DISK/ BASIC.SYSTEM** and pressing Return for the COPY PATHNAME prompt. Type **/TBA1/BASIC.SYSTEM**, then press Return for the

TO PATHNAME prompt. Press Return again, and the file BASIC.SYSTEM will be placed into memory. When prompted, remove the *ProDOS User's Disk* and reinsert your newly formatted disk, then press Return. A copy of the file BASIC.SYSTEM will be placed onto your newly formatted disk. When you see the message COPY COMPLETE, press Esc to return to the File Commands menu. Follow the above instructions to copy the files PRODOS and BASIC.SYSTEM to your second disk (the one with the volume name TBA2).

5. Remove the formatted disk and insert the *Third Book of Apple* disk into the drive. Be sure that the side labeled PRODOS is facing upward. Type L for LIST PRODOS DIRECTORY. For the DIRECTORY PATHNAME prompt, type /AAP3BK and press Return. When the directory appears, write down all the filenames you see and then press Return. Now write down any additional filenames you see and press Esc.
6. Type C for COPY FILES. For the COPY PATHNAME prompt, type /AAP3BK/MENU, then press Return. For the TO PATHNAME prompt, type /TBA1/MENU, then press Return. Press Return again, and the file TUGAWAR will be placed into memory. When prompted, remove the *Third Book of Apple* disk and reinsert the TBA1 disk, then press Return to start the copy. When you see the message COPY COMPLETE, remove the newly formatted disk and reinsert the *Third Book of Apple* disk and go to the next filename you wrote down earlier. Repeat this procedure for all the files you wrote down, being careful not to mix up the disks and to keep the ProDOS side of the *Third Book of Apple* disk facing upward. To avoid typing the pathname and filename on each file, simply move the cursor over to the end of the prompt and type in the filenames only.
7. When you see the message

FILE NAME TOO LARGE

it's time to switch disks. Remove the TBA1 disk and replace it with the TBA2 disk. Continue with your copying, substituting TBA2 for TBA1 in the pathname. Be sure to copy MENU, AASET, and HROUT to both disks.

8. To use either disk, type RUN MENU at the] prompt and press Return.

Apple IIc ProDOS Users

1. Insert your *System Utilities* disk and turn on the computer.
2. From the System Utilities menu, type 6, then press Return, to select the Format a Disk option.
3. Press Return to select the built-in drive, then Return again to select ProDOS format.
4. Remove the *System Utilities* disk and insert a blank, unformatted disk in the drive.
5. Type in a name for your new disk volume, followed by Return.
6. When formatting is complete, press Esc to return to the main menu. Format another disk in the same manner, using a different name for the second disk volume. (You'll need two disks of your own to hold all the files on the *Third Book of Apple* disk.)
7. Remove the newly formatted disk and insert your *System Utilities* disk in the drive (it will be the source disk).
8. Type 1, then Return, to select the Copy Files option. Press Return twice more to specify the built-in drive for both the source and destination disks.
9. Press Return to copy some of the files on the disk.
10. Move the cursor up or down until the < > indicator is on the file called PRODOS.
11. Mark this file for copying by pressing the cursor-right key.
12. Mark the file BASIC.SYSTEM for copying by using the method explained in steps 10 and 11.
13. Press Return. The PRODOS file will be copied into memory.
14. When prompted, remove the *System Utilities* disk and insert the first newly formatted disk in the drive (it is the destination disk). Part of the PRODOS file will now be copied to your disk.
15. When prompted, remove your disk and again insert the *System Utilities* disk, then press Return. The remainder of the PRODOS file will be copied into memory.
16. When prompted, remove the *System Utilities* disk, insert your newly formatted disk, and press Return.
17. Repeat steps 15 and 16 to copy the BASIC.SYSTEM file to your disk.
18. Press Esc to return to the System Utilities main menu. Follow steps 7-17 to place the files PRODOS and BASIC.SYSTEM onto your second formatted disk.

19. Remove your disk and insert the *Third Book of Apple* disk into the drive. Be sure you have the side of the disk labeled ProDOS facing upward.
20. Type 1, then Return, to select the Copy Files option. Then press Return twice more to specify the built-in drive for both the source and destination disks. Select <ALL> by moving the cursor right, then pressing Return.
21. You will be prompted to switch back and forth between the *Third Book of Apple* disk (the source disk) and your disk containing the PRODOS and BASIC.SYSTEM files. Numerous swaps will be required to complete the copying, so be careful not to get the disks mixed up. Also, be sure that you don't turn over the *Third Book of Apple* disk; always keep the side with the ProDOS label facing upward when you insert it into the drive.
22. When you see the message which begins

Disk Is Full

- it's time to switch the destination disk. From this point, use the TBA2 disk. Continue with your copying.
23. You'll need to copy MENU, AASET, and HROUT to TBA2. Press Return to return to Copy Files. Press Return twice more to specify the built-in drive. Select <SOME> and press Return. Use the arrow keys to select the files MENU, AASET, and HROUT. Press Return and follow the screen prompts.
 24. To use either disk, type RUN MENU at the] prompt and press Return.

Index

- "Advanced Mousification" program 264-71
- Alquerque board game 33
- ampersand (&) 297
- animation 201-5
- "Animator2" graphics file program 205-9
- "Apple Animator" program v, 201-15
 - source code 209-15
- "Apple Automatic Proofreader" program 321-24
- "Apple Disk Booster" program 292-95
- "Apple Disk Duper" program 313-14
- Apple hi-res screen dump 222-26
- "Apple Keyboard Customizer" program 296-301
- "Apple MLX" program 18, 40, 173, 325-30
- Apple screen editing 286-89
- "Applesoft List Enhancer" program 315-16
- "Apple SpeedCalc" program 77-131
 - command summary 94
 - cursor movement 79-80
 - data formats 88-89
 - DOS 3.3 version 78, 94-113
 - editing 86-87
 - formula data 84-86
 - functions 85-86
 - keyboard commands 80-82
 - macro editing 89-90
 - memory available 90-91
 - numeric data 82-83
 - preparing 78-79
 - printing 92
 - ProDOS version 78, 114-32
 - recalculation 87-88
 - text data 84
- "Apple II Mouse Demonstration" program 259-64
- "Apple II Pull-Down Menus" program 246-49
- applications 77-197
- "ApWriter" program 144-52
 - command summary 149
 - margins 147
 - printer compatibility 148
 - tabs 146-47
- ASCII code 297
- astronomy 57-61
- auditory feedback, keyboard 283-85
- "BASIC Line Editor" program 286-91
 - commands 287-88
- BASIC memory, relocating 18, 78
- BRUN command 18, 41, 174
- Caps Lock key 325
- cell, spreadsheet 79, 80
- changing entries, "Apple SpeedCalc" and 83
- checksum 322
- CHR\$(4) function 144
- command summary, "Apple SpeedCalc" 94
- command summary, "ApWriter" 149
- COMPUTE!'s *Third Book of Apple* disk 331-39
- Ctrl (control) key 322
- cursor movement, "Apple SpeedCalc" 79-80
- cursor movement, "Dr. Disk" 175
- customized keyboard, saving through reboot 299
- data formats, "Apple SpeedCalc" 88-89
- DATA statement 11, 320
- Dazzle Draw program 216
- disk
 - backing up 313
 - COMPUTE!'s *Third Book of Apple* 331-39
 - directory (DOS 3.3) 178, 180-83
 - double-DOS 185-88
 - editing 173-88
 - operations, "Apple SpeedCalc" 91-92
- display, 40/80 column 145
- display modes 145
- "Dr. Disk" program 173-98
- DOS 3.3 90, 173, 284, 292, 319, 332-34
 - and ProDOS, on one disk 185-88
 - commands, "Dr. Disk" 174-76
- double-DOS disk 185-88
 - cursor movement 175
 - discovering forgotten password with 177-79
 - DOS 3.3 source code 188-92
 - printer slot and 174
 - ProDOS source code 193-97
 - undeleting files with 179-85
- "DUMP Example" program 226
- "DUMP" program 222-26
- editing, "Apple SpeedCalc" 86-87
- education 53-73
- escape mode editing 275-79, 286
- extra tracks, formatting on disk 292-94
- Fantavision program 216
- Filer ProDOS program 184
- file structure, ProDOS 184-85

formula data, "Apple *SpeedCalc*" 84-86
 functions, "Apple *SpeedCalc*" 85-86
 graphics 201-25
 Halley's Comet 60-61
 "Help Screen Editor" program 302-7
 help screens 302-4
 hexadecimal notation 175
 "Hickory, Dickory, Dock" program 53-56
 high-resolution graphics page 311
 "High Rise" program 18-32
 "Hi-Res Graphics Aid" program 216-21
 HTAB function 144
 INPUT statement 251
 joystick 40, 253-54, 258
 "Joystick Modifications" program 274
 keyboard, customized, saving through reboot 299
 keyboard, customizing *v*, 296-99
 keyboard commands, "Apple *SpeedCalc*" 80-82
 keyboard reference (chart), "Apple *SpeedCalc*" 81
 "Keynote" program 283-85
 Macintosh computer *v*, 229, 241
 macro editing, "Apple *SpeedCalc*" and 89-90
 margins, "ApWriter" 147
 mathematical operators, "Apple *SpeedCalc*" and 85
 "Memo Diary" program 133-43
 running first time 133-34
 memory available with "Apple *SpeedCalc*" 90-91
 memory mapping 251
 memory partitioning 308-11
 menus, pull-down *v*, 241-44
 "Miami Ice" program *v*, 39-50
 mouse 243
 adding to Applesoft programs 250-59
 editing functions 256-58
 interface card 250
 sensitivity 253
 Mousetext special graphics characters 145
 "MOUSEY File" program 271-73
 "Mouser" program 275-79
 "MOVE Routine" program 244-46
 "MultiMemory" program *v*, 308-12
 numeric data, "Apple *SpeedCalc*" and 82-83
 Open Apple key 80-82
 emulating on Apple II and Apple II+ 82
 paddle 40
 partitions 308-11
 password, forgotten, discovering with "Dr. Disk" 177-79
 pattern recognition 70-71
 personal finance 153-61
 pivot date, "Memo Diary" and 135-36
 PR#1 command 144
 printer
 activating 144
 compatibility, "ApWriter" and 148
 slot, "Dr. Disk" and 174
 printing, "Apple *SpeedCalc*" 92
 PRINT statement 144
 ProDOS 91, 173, 284, 313, 319, 334-39
 and DOS 3.3, on one disk 185-88
 commands, "Dr. Disk" 176-77
 file structure, ProDOS 184-85
 program listings, enhancing 315
 pull-down menus *v*, 241-44
 "Puzzler" program 70-74
 modifying 71
 recalculation, "Apple *SpeedCalc*" and 87-88
 recreation 3-49
 ROM editor 286
 screen, "Apple *SpeedCalc*" 79
 screen editing 275-79
 "Screen Editor Loader" program 305
 sector, disk 175
 "Skyscape" program *v*, 57-69
 "SpeedScript File Converter" program 77, 93
 "SpeedScript" word processor 77, 93
 spreadsheet, electronic *v*, 77-93
 "Switchbox" program 7-17
 tabs, "ApWriter" 146-47
 text data, "Apple *SpeedCalc*" 84
 "TEXT Filemaker" program 274
 time, telling 53
 track 175
 "Tug-A-War" program 3-6
 typing in programs 319-20
 undeleting a file under DOS 3.3, "Dr. Disk" and 179-83
 undeleting a file under ProDOS, "Dr. Disk" and 183-85
 uppercase 319
 Volume Bit Map 184, 187
 Volume Table of Contents. *See* VTOC
 VTAB function 144
 VTOC 179-83, 187, 292-93
 wedge 285
 windows *v*, 229-40
 "Windows in BASIC" program 238-40
 "Windows" program 229-40
 "Witching Hour, The" program 33-38
 word processors, uses and limitations of 144
 "Your Personal Ledger" program 153-72

To order your copy of *COMPUTE!'s Third Book of Apple Disk*, call our toll-free US order line: 1-800-346-6767 (in NY 212-887-8525) or send your prepaid order to:

COMPUTE!'s Third Book of Apple Disk
COMPUTE! Publications
P.O. Box 5038
F.D.R. Station
New York, NY 10150

All orders must be prepaid (check, charge, or money order). NC residents add 4.5% sales tax.

Send _____ copies of *COMPUTE!'s Third Book of Apple Disk* at \$12.95 per copy.

Subtotal \$_____

Shipping and Handling: \$2.00/disk \$_____

Sales tax (if applicable) \$_____

Total payment enclosed \$_____

☐ Payment enclosed

☐ Charge ☐ Visa ☐ MasterCard ☐ American Express

Acct. No. _____ Exp. Date _____
(Required)

Name _____

Address _____

City _____ State _____ Zip _____

Please allow 4-5 weeks for delivery.

To order your copy of CONQUEST, West Book or Paper Back,
call our hotline 1-800-426-6747 on the 212-
527-5122 or send your 1-400-512-5122

CONQUEST, P.O. Box 5122
CONQUEST Publications
P.O. Box 5122

New York, NY 10122

Attention: Please send me a copy of CONQUEST, P.O. Box 5122,
New York, NY 10122

Name _____
Address _____

City _____
State _____
Zip _____
Country _____
Telephone _____

Payment enclosed _____
Charge to my MasterCard or American Express _____

Card No. _____
Exp. Date _____

Name _____

Address _____

City _____

State _____

Zip _____

COMPUTE! Books

Ask your retailer for these **COMPUTE! Books** or order directly from **COMPUTE!**.

Call toll free (in US) **1-800-346-6767** (in NY 212-887-8525) or write COMPUTE! Books, P.O. Box 5038, F.D.R. Station, New York, NY 10150.

Quantity	Title	Price*	Total
_____	Becoming a MacArtist (80-9)	\$17.95	_____
_____	COMPUTE!'s Apple Games for Kids (91-4)	\$12.95	_____
_____	COMPUTE!'s First Book of Apple (69-8)	\$12.95	_____
_____	COMPUTE!'s Guide to Telecomputing on the Apple (98-1)	\$ 9.95	_____
_____	COMPUTE!'s Kids and the Apple (76-0)	\$12.95	_____
_____	Easy BASIC Programs for the Apple (88-4)	\$14.95	_____
_____	MacTalk: Telecomputing on the Macintosh (85-X)	\$14.95	_____
_____	SpeedScript: The Word Processor for Apple Personal Computers (000)	\$ 9.95	_____
_____	The Apple IIc: Your First Computer (001)	\$ 9.95	_____
_____	Apple Machine Language for Beginners (002)	\$14.95	_____
_____	COMPUTE!'s Second Book of Apple (008)	\$12.95	_____
_____	MacOffice: Using the Macintosh for Everything (006)	\$14.95	_____
_____	MacIdeas (015-7)	\$14.95	_____
_____	Using Your Macintosh: Beginning Microsoft BASIC and Applications (021-1)	\$16.95	_____
_____	Apple II Applications: 40 Programs for Your Apple (016-5)	\$14.95	_____
_____	Advanced Macintosh BASIC Programming (030-0)	\$16.95	_____
_____	COMPUTE!'s Third Book of Apple (063-7)	\$14.95	_____

*Add \$2.00 per book for shipping and handling.
Outside US add \$5.00 air mail or \$2.00 surface mail.

NC residents add 4.5% sales tax. _____

Shipping & handling: \$2.00/book _____

Total payment _____

All orders must be prepaid (check, charge, or money order).

All payments must be in US funds.

☐ Payment enclosed.

Charge ☐ Visa ☐ MasterCard ☐ American Express

Acct. No. _____ Exp. Date _____
(Required)

Name _____

Address _____

City _____ State _____ Zip _____

*Allow 4-5 weeks for delivery.

Prices and availability subject to change.

Current catalog available upon request.



An Extraordinary Collection

Some of the best Apple software you'll find anywhere comes from COMPUTE! Publications.

High-quality, ready-to-type-in programs from *COMPUTE!* magazine and *COMPUTE!'s Apple Applications Special* have been chosen for our third Apple anthology, *COMPUTE!'s Third Book of Apple*. Inside you'll find exciting arcade games, impressive home and business applications, colorful graphics creators, useful programming utilities, and entertaining educational software.

- *SpeedCalc*, an all machine language spreadsheet that rivals commercial software in ease of use, power, and features.
- "High Rise," an arcade game of running, dodging, and falling. Perhaps our best Apple game yet.
- "Hickory, Dickory, Dock" helps youngsters tell time with a colorful clock and hands that move.
- "Apple Animator" puts you in charge of your own animation studio.
- Three programs magically make your Apple II act like a Macintosh, adding pull-down menus, windows, and mouse movement.
- And programming utilities and tutorials that customize the keyboard, simplify editing, and duplicate disks at high speed.

Understandable instructions, thoroughly tested programs, and error-checking aids that simplify typing in our Apple listings make *COMPUTE!'s Third Book of Apple* the best software bargain around.

All the programs in this book are available on a companion disk. See the coupon in the back for details.